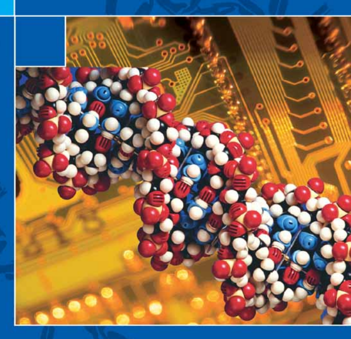
Pavel A. Pevzner

Collection IRIS
dirigée par Nicolas Puech



Bio-informatique moléculaire

Une approche algorithmique

Traduction : Delphine Hachez



Bio-informatique moléculaire

Une approche algorithmique

Springer

Paris
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Tokyo

Pavel A. Pevzner

Bio-informatique moléculaire

Une approche algorithmique

Traduit de l'anglais par
Delphine Hachez



Pavel A. Pevzner
Department of Computer Science & Engineering,
APM 3132
University of California, San Diego
La Jolla, CA 92093-0114

Traduit de l'édition anglaise par Delphine Hachez :
Computational Molecular Biology: an algorithmic approach

ISBN-10 : 2-287-33908-6 Springer Paris Berlin Heidelberg New York

ISBN-13 : 978-2-287-33908-0 Springer Paris Berlin Heidelberg New York

Imprimé en France

Springer-Verlag France est membre du groupe Springer Science + Business Media

SPIN : 11748 809

Maquette de couverture : Jean-François Montmarché

Préface

En 1985, alors que je cherchais du travail à Moscou, en Russie, j'ai été confronté à un choix difficile. D'un côté, une prestigieux institut d'ingénierie électrique m'offrait un poste de chercheur en combinatoire appliquée ; de l'autre, le centre russe de biotechnologie NIIGENETIKA, situé dans la banlieue de Moscou, était en train de former un groupe de bio-informatique. Ce second poste était rémunéré environ moitié moins que l'autre et n'offrait même pas le « zakaz » hebdomadaire, ce colis alimentaire qui représentait l'avantage le plus important accordé à un employé, à une époque où les rayonnages des magasins de Moscou étaient vides. Je ne sais toujours pas à quel genre de recherche classée secrète s'adonnaient les gens de l'institut d'ingénierie électrique, puisqu'ils n'avaient pas le droit de m'en parler avant que je ne signe mon contrat. En revanche, Andrey Mironov, du NIIGENETIKA, passa quelques heures à me parler des problèmes algorithmiques d'une nouvelle discipline futuriste appelée *bio-informatique moléculaire* et je pris ma décision. Je ne l'ai jamais regrettée, bien que, pendant un certain temps, je dus compléter mes revenus au NIIGENETIKA en ramassant les bouteilles vides dans les gares de Moscou, l'un des rares moyens légaux de gagner un peu d'argent supplémentaire dans le Moscou d'avant la perestroïka.

La bio-informatique était une discipline nouvelle pour moi et j'ai passé des week-ends entiers à la bibliothèque Lénine de Moscou, le seul endroit où il était possible de trouver des articles de bio-informatique. L'unique livre disponible à l'époque était le très classique *Time Warps, String Edits and Biomolecules : The Theory and Practice of Sequence Comparison* de Sankoff et Kruskal. Comme les photocopieuses Xerox étaient quasiment inexistantes à Moscou en 1985, j'ai presque copié ce livre page par page dans mes cahiers. Six mois plus tard, j'ai réalisé que j'avais lu l'intégralité (ou presque) de tous les articles de bio-informatique du monde. Ma foi, ce n'était pas un grand exploit : la majeure partie d'entre eux étaient écrits par les « pères fondateurs » de la bio-informatique moléculaire, David Sankoff et Michael Waterman, et je n'avais qu'à feuilleter une demi-douzaine de journaux. Durant les sept années suivantes, j'allais à la bibliothèque une fois par mois et je lisais tout ce qui avait été publié dans ce domaine. Cette situation ne dura pas longtemps. Dès 1992, je me suis rendu compte que l'explosion avait débuté : je n'avais plus le temps de lire toutes les publications de bio-informatique.

Comme certains journaux n'étaient pas disponibles à la bibliothèque Lénine, je les demandais à des scientifiques étrangers et bon nombre d'entre eux étaient assez aimables pour m'envoyer leurs *preprints*. En 1989, je reçus un lourd paquet de Michael Waterman avec une douzaine de manuscrits en préparation. L'un d'eux formulait un problème ouvert que je résolus ; j'envoyai ma solution à Mike, sans trop me préoccuper des preuves. Mike m'avoua plus tard que la lettre était écrite dans un style très « anglais russe » impossible à comprendre, mais qu'il était surpris que quelqu'un fût capable de lire son propre travail jusqu'à l'endroit où se trouvait le problème ouvert. Peu de temps après, Mike m'invita à travailler avec lui à l'Université de Californie du Sud et, en 1992, je donnai mon premier cours de bio-informatique.

Ce livre est fondé sur le cours de *bio-informatique moléculaire* que j'ai donné tous les ans dans le département d'informatique de l'Université de Pennsylvanie (de 1992 à 1995), puis dans le département de mathématiques de l'Université de Californie du Sud (de 1996 à 1999). Il est destiné aux étudiants de licence et de maîtrise d'informatique ou de mathématiques, ainsi qu'aux étudiants de D.E.U.G. de bon niveau. Certaines parties du livre présenteront aussi de l'intérêt pour les spécialistes de biologie moléculaire qui s'intéressent à la bio-informatique. J'espère également que cet ouvrage sera utile aux professionnels de bio-informatique.

La raison d'être de ce livre est de présenter des idées algorithmiques de bio-informatique et de montrer la façon dont elles sont reliées à la biologie moléculaire et à la biotechnologie. Pour y parvenir, cet ouvrage possède une composante substantielle de « bio-informatique sans formule » qui présente une motivation biologique et des idées informatiques de façon simple. Cette présentation simplifiée de biologie et d'informatique vise à rendre ce livre accessible aux informaticiens qui découvrent ce nouveau domaine ainsi qu'aux biologistes n'ayant pas un bagage suffisant pour aborder des techniques informatiques plus évoluées. Par exemple, le chapitre *recherche en génétique* décrit de nombreux résultats informatiques associés à la recherche du gène de la mucoviscidose et formule des problèmes informatiques que ces résultats ont motivés. Chaque chapitre comporte une introduction qui décrit les idées informatiques et biologiques sans formule. Ce livre se concentre sur les idées informatiques plutôt que sur les détails des algorithmes et un gros effort a été fait pour les présenter de façon simple. Évidemment, pour parvenir à un tel exposé, on est obligé de masquer certains détails informatiques et biologiques au risque d'être critiqué pour cette « vulgarisation » de bio-informatique. Une autre caractéristique de cet ouvrage est que la dernière partie de chaque chapitre décrit brièvement les récents développements importants qui sont en dehors du corps du chapitre.

Dans le département d'informatique, les cours de bio-informatique débutent souvent par une introduction de deux à trois semaines que l'on pourrait intituler « la biologie moléculaire pour les nuls ». Si j'en crois mon expérience, l'intérêt des étudiants en informatique (qui, en règle générale, n'ont aucune connaissance en biologie) diminue rapidement s'ils sont confrontés à une introduction à la biologie qui ne présente aucun lien avec l'informatique. Il se produit

le même phénomène avec les biologistes si on leur expose des algorithmes sans lien avec les problèmes biologiques réels. J'ai trouvé très important d'introduire la biologie et les algorithmes de façon simultanée, afin de susciter et de préserver l'intérêt des étudiants. Le chapitre *recherche en génétique* atteint ce but, bien qu'il présente volontairement une vision simplifiée de la biologie et des algorithmes. J'ai aussi trouvé que certains bio-informaticiens n'avaient pas une idée bien précise des liens entre les différents domaines de la bio-informatique. Par exemple, les chercheurs qui travaillent sur la prédiction génétique ont parfois des connaissances limitées concernant les algorithmes de comparaison de séquences. J'ai donc tenté de mettre en exergue les liens existant entre les idées informatiques et les différents domaines de la bio-informatique moléculaire.

Ce livre couvre à la fois les secteurs les plus récents de la bio-informatique et ceux plutôt anciens. Par exemple, les matériaux présents dans le chapitre *protéomique informatique* et la majeure partie de ceux des chapitres *réarrangements génomiques*, *comparaison de séquences* et *puces à ADN* n'avaient jamais été abordés dans un livre jusqu'à maintenant. Par ailleurs, les sujets comme ceux abordés dans le chapitre *cartographie de restriction* sont plutôt passés de mode et décrivent des approches expérimentales qui sont tombées en désuétude. J'ai tout de même inclus ces notions informatiques un peu anciennes pour deux raisons. Tout d'abord, elles expliquent aux néophytes l'histoire des idées dans ce domaine et les avertit que les sujets les plus chauds de la bio-informatique évoluent très rapidement. En outre, ces idées informatiques ont souvent une seconde vie dans différents domaines d'application. Par exemple, des techniques presque oubliées de cartographie de restriction trouvent une nouvelle utilité dans le domaine très à la mode de la protéomique informatique. Il existe de nombreux autres exemples de ce genre (par exemple, certaines idées relatives au séquençage par hybridation sont actuellement utilisées pour l'assemblage shotgun à grande échelle) et je pense qu'il est important de montrer à la fois les anciennes et les nouvelles approches informatiques.

Je tiens aussi à dire quelques mots sur un compromis qui a été fait dans ce livre entre les parties expérimentales et théoriques. Il est certain que les biologistes du XXIe siècle devront connaître des éléments de mathématiques discrètes et d'algorithmique — ils devront au moins être capables de formuler les problèmes algorithmiques motivés par leur recherche. En bio-informatique, la formulation adéquate des problèmes biologiques est probablement la composante la plus difficile de la recherche, au moins aussi difficile que leur résolution. Comment pouvons-nous apprendre aux étudiants à formuler des problèmes biologiques en termes informatiques ? Comme je n'en sais rien, je préfère vous offrir une histoire en guise de réponse.

Il y a vingt ans, après avoir obtenu mon diplôme universitaire, j'ai mis une annonce pour proposer mes services et faire du « conseil mathématique » à Moscou. Mes clients étaient majoritairement des doctorants dans différentes matières appliquées qui n'avaient pas un bagage mathématique suffisant et qui espéraient obtenir une aide pour leur thèse (ou, du moins, ses composantes mathématiques). Je me suis retrouvé face à une vaste collection de sujets allant

de « l'optimisation du parc de l'équipement de déblaiement de la neige dans un aéroport » à « la mise en place d'un programme de livraison de fournisseurs ». Dans tous ces projets, la partie la plus difficile était de comprendre ce qu'était le problème informatique et de le formuler ; la solution s'obtenait par application directe de techniques connues.

Jamais je n'oublierai un visiteur, quarante ans, poli, bien bâti. Contrairement aux autres, celui-ci vint avec une équation différentielle à résoudre, plutôt qu'avec son domaine de recherche. Au début, j'étais content mais par la suite, il s'avéra que cette équation n'avait aucun sens. La seule façon de comprendre ce qu'il fallait faire était de revenir au problème expérimental d'origine et d'en déduire une nouvelle équation. Le visiteur hésita mais, comme c'était le seul moyen qu'il avait d'obtenir son doctorat, il commença à me révéler certains détails concernant son domaine de recherche. À la fin de la journée, j'avais compris qu'il s'intéressait à l'atterrissage d'objets sur une plateforme peu solide. J'avais également deviné pourquoi il ne m'avait jamais donné son numéro de téléphone : il s'agissait d'un officier qui travaillait sur une recherche classée secrète. La plateforme mal affermie était un navire et les objets qui atterrissaient étaient des avions. Je suis convaincu que la révélation de ce secret, vingt ans plus tard, ne brisera pas sa carrière militaire.

La nature est encore moins ouverte à la formulation de problèmes biologiques que cet officier. En outre, certains d'entre eux, lorsqu'ils sont formulés correctement, présentent de nombreuses options qui peuvent parfois masquer ou travestir les idées informatiques. Comme il s'agit ici d'un livre traitant de ces dernières plutôt que des détails techniques, j'ai volontairement utilisé des formulations simplifiées qui permettent de présenter les idées de façon claire. Ceci peut donner l'impression que cet ouvrage est trop théorique, mais je ne connais pas d'autre moyen d'enseigner des idées informatiques en biologie. En d'autres termes, avant de faire atterrir de vrais avions sur de vrais navires, les étudiants doivent apprendre comment faire atterrir des maquettes d'avions sur des maquettes de bateaux.

J'aimerais insister sur le fait que cet ouvrage n'a pas la prétention de couvrir uniformément tous les secteurs de la bio-informatique. Certes, le choix des sujets a été influencé par mes propres goûts et recherches. Quelques grands domaines de la bio-informatique ne sont pas abordés — notamment les statistiques ADN, la cartographie génétique, l'évolution moléculaire, la prédiction de la structure protéique et la génomique fonctionnelle. Chacun de ces domaines mérite un ouvrage à part entière ; certains ont d'ailleurs déjà été écrits. Par exemple, Waterman, 1995 [357] est une excellente référence pour les statistiques ADN. Gusfield, 1997 [145] présente de nombreux algorithmes de chaînes et Salzberg *et al.*, 1998 [296] contient quelques chapitres qui couvrent la prédiction de la structure protéique. Durbin *et al.*, 1998 [93] et Baldi et Brunak, 1997 [24] sont des livres plus spécialisés qui se focalisent sur les modèles de Markov. Baxevanis et Ouellette, 1998 [28] est un excellent guide pratique de bio-informatique qui se consacre davantage aux applications des algorithmes qu'aux algorithmes eux-mêmes.

J'aimerais remercier quelques personnes qui m'ont appris différents aspects de bio-informatique moléculaire. Andrey Mironov m'a appris que le bon sens est peut-être l'ingrédient le plus important de toute recherche appliquée. Mike Waterman est un fabuleux enseignant, à l'époque où je suis parti de Moscou pour Los Angeles, que ce soit en science ou dans la vie. En particulier, il m'a appris avec beaucoup de patience que chaque article devait subir une douzaine d'itérations avant d'être prêt à être publié. Bien que cette règle retarde la publication de ce livre de quelques années, je l'enseigne scrupuleusement à mes étudiants. Mes anciens étudiants Vineet Bafna et Sridhar Hannenhalli ont été assez aimables pour m'apprendre ce qu'ils savaient et me rejoindre dans de difficiles projets à long terme. J'aimerais également remercier Alexander Karzanov, qui m'a enseigné l'optimisation combinatoire, y compris les idées qui ont été les plus utiles dans mes recherches en bio-informatique.

Je voudrais remercier mes collaborateurs et co-auteurs : Mark Borodovsky, avec qui j'ai travaillé sur les statistiques ADN et qui m'a convaincu en 1985 que la bio-informatique avait un grand avenir ; Earl Hubbell, Rob Lipshutz, Yuri Lysov, Andrey Mirzabekov et Steve Skiena, mes collègues pour la recherche sur les puces à ADN ; Eugene Koonin, avec qui j'ai essayé d'analyser des génomes complets, avant le séquençage du premier génome bactérien ; Norm Arnheim, Mikhail Gelfand, Melissa Moore, Mikhail Roytberg et Sing-Hoi Sze, mes collègues en recherche génétique ; Karl Clauser, Vlado Dancik, Maxim Frank-Kamenetsky, Zufar Mulyukov et Chris Tang, mes collaborateurs en protéomique informatique ; enfin, Eugene Lawler, Xiaoqiu Huang, Webb Miller, Anatoly Vershik et Martin Vingron, mes collègues en comparaison de séquences.

Je suis également reconnaissant à de nombreux collègues d'avoir discuté avec moi de différents aspects de bio-informatique moléculaire ; directement ou indirectement, ils ont influencé la rédaction de cet ouvrage : Ruben Abagyan, Nick Alexandrov, Stephen Altschul, Alberto Apostolico, Richard Arratia, Ricardo Baeza-Yates, Gary Benson, Piotr Berman, Charles Cantor, Radomir Crkvenjakov, Kun-Mao Chao, Neal Copeland, Andreas Dress, Radoje Drmanac, Mike Fellows, Jim Fickett, Alexei Finkelstein, Steve Fodor, Alan Frieze, Dmitry Frishman, Israel Gelfand, Raffaele Giancarlo, Larry Goldstein, Andy Grigoriev, Dan Gusfield, David Haussler, Sorin Istrail, Tao Jiang, Sampath Kannan, Samuel Karlin, Dick Karp, John Kececioglu, Alex Kister, George Komatsoulis, Andrzey Konopka, Jenny Kotlerman, Leonid Kruglyak, Jens Lagergren, Gadi Landau, Eric Lander, Gene Myers, Giri Narasimhan, Ravi Ravi, Mireille Regnier, Gesine Reinert, Isidore Rigoutsos, Mikhail Roytberg, Anatoly Rubinov, Andrey Rzhetsky, Chris Sander, David Sankoff, Alejandro Schaffer, David Searls, Ron Shamir, Andrey Shevchenko, Temple Smith, Mike Steel, Lubert Stryer, Elizabeth Sweedyk, Haixi Tang, Simon Tavarè, Ed Trifonov, Tandy Warnow, Haim Wolfson, Jim Vath, Shibu Yooseph et les autres.

Travailler avec Bob Prior et Michael Rutterof de MIT Press a été un réel plaisir. Je remercie également Amy Yeager, qui a édité ce livre, Mikhail Mayofis, qui a réalisé la couverture de la version anglaise, et Oksana Khleborodova,

qui a illustré les différentes étapes de l'algorithme de prédiction génétique. Je voudrais aussi remercier ceux qui ont soutenu mes recherches : le Ministère de l'énergie, l'Institut national de la santé et la Fondation nationale pour la science.

Enfin et surtout, j'adresse tous mes remerciements à Paulina et Arkasha Pevzner, qui ont été assez gentilles pour rester calmes et tolérer mon manque de disponibilité pendant que je rédigeais cet ouvrage.

Table des matières

Chapitre 1

Recherche en génétique

1.1 Introduction

La mucoviscidose est une maladie mortelle associée à des infections respiratoires récurrentes et à des sécrétions anormales. Cette maladie est diagnostiquée chez un enfant sur deux mille cinq cents. Un individu de type caucasien sur vingt-cinq possède le gène responsable de la mucoviscidose ; les enfants qui héritent des gènes de leurs *deux* parents sont atteints.

Dans le milieu des années 80, les biologistes ne savaient rien sur le gène qui est à l'origine de la mucoviscidose et il n'existait pas de diagnostic prénatal fiable. Le meilleur espoir de guérison pour de nombreuses maladies génétiques repose en fait sur la découverte des gènes défectueux. Pour le gène de la mucoviscidose, la recherche débuta au début des années 80 et, en 1985, trois groupes de scientifiques prouvèrent simultanément et indépendamment que le gène de la mucoviscidose se trouvait sur le septième chromosome. En 1989, la recherche se restreignait à un petit secteur du septième chromosome et le gène de la mucoviscidose, long de 1480 acides aminés, fut découvert. Cette découverte conduisit à des diagnostics médicaux efficaces et à la promesse d'une possible thérapie pour lutter contre la mucoviscidose. La recherche génétique sur cette maladie est un travail ardu entrepris à la fin des années 80. Depuis, des milliers de gènes ayant une importance d'ordre médical ont été découverts et la recherche de nombreux autres est actuellement en cours. La recherche génétique comporte beaucoup de problèmes informatiques, dont certains sont passés en revue dans ce chapitre.

1.2 Cartographie génétique

Tout comme les cartographes dressaient des cartes du monde antique, les biologistes ont péniblement dressé la cartographie de l'ADN humain durant les trois dernières décennies du XXe siècle. Le but est de déterminer la position

des gènes sur les différents chromosomes, afin de comprendre la géographie du génome.

Lorsque la recherche sur le gène de la mucoviscidose a commencé, les scientifiques ne possédaient aucun indice quant à la nature du gène ou à sa position dans le génome. La recherche génétique débute habituellement avec la *cartographie génétique*, qui fournit une localisation approximative du gène sur l'un des chromosomes humains (normalement, à l'intérieur d'un secteur formé de quelques millions de nucléotides). Pour comprendre les problèmes informatiques liés à la cartographie génétique, on utilise un modèle très simplifié de cartographie génétique pour des *robots* mono-chromosomiques. Chaque robot possède n gènes (dans un ordre inconnu) et chaque gène peut être dans l'un des états 0 ou 1, aboutissant à deux *phénotypes* (traits physiques) *rouge* ou *marron*. Si l'on suppose que n vaut 3 et que les trois gènes du robot définissent la couleur de ses cheveux, de ses yeux et de ses lèvres, alors 000 correspond à un robot *tout rouge* (cheveux rouges, yeux rouges, lèvres rouges), tandis que 111 donne un robot *tout marron*. Bien que nous puissions observer les phénotypes des robots (i.e., la couleur de leurs cheveux, de leurs yeux et de leurs lèvres), on ne connaît pas l'ordre des gènes dans leur génome. Heureusement, les robots peuvent avoir des enfants et ceci nous aide à construire leurs cartes génétiques.

Un enfant des robots $m_1 \dots m_n$ et $p_1 \dots p_n$ est soit un robot $m_1 \dots m_i$ $p_{i+1} \dots p_n$, soit un robot $p_1 \dots p_i m_{i+1} \dots m_n$ pour une *position de recombinaison* i, avec $0 \leq i \leq n$. Chaque paire de robots peut avoir $2(n+1)$ différentes sortes d'enfants (certains peuvent être identiques) et la probabilité que la recombinaison ait lieu en position i est égale à $\frac{1}{(n+1)}$.

Problème de la cartographie génétique Étant donnés les phénotypes d'un grand nombre d'enfants de robots tout rouge et tout marron, trouver l'ordre des gènes chez les robots.

L'analyse des fréquences de différentes *paires* de phénotypes nous permet d'en déduire l'ordre des gènes. Il faut calculer la probabilité p qu'un enfant issu d'un robot tout rouge et d'un robot tout marron n'ait pas les cheveux et les yeux de la même couleur. Si le gène des cheveux et celui des yeux sont consécutifs dans le génome, alors la probabilité de recombinaison entre ces gènes est de $\frac{1}{n+1}$. Si le gène des cheveux et celui des yeux ne sont pas consécutifs, alors la probabilité qu'un enfant n'ait pas les cheveux et les yeux de la même couleur vaut $p = \frac{i}{n+1}$, où i est la *distance* entre ces gènes dans le génome. Mesurer p dans la population des enfants nous aide à estimer les distances entre les gènes, à trouver leur ordre et à reconstruire la carte génétique.

Dans le monde des robots, le chromosome d'un enfant est constitué de deux fragments : l'un est issu du robot mère et l'autre du robot père. Dans un modèle de recombinaison plus exact (mais encore irréaliste), le génome d'un enfant est défini comme la mosaïque d'un nombre arbitraire de fragments des génomes de la mère et du père, de sorte que l'on a : $m_1 \dots m_i p_{i+1} \dots p_j m_{j+1} \dots m_k p_{k+1} \dots$ Dans ce cas, la probabilité de recombinaison entre deux gènes est proportion-

nelle à la distance entre ceux-ci et, comme précédemment, plus les gènes sont éloignés, plus les recombinaisons entre eux sont fréquentes. Si deux gènes sont très proches l'un de l'autre, les recombinaisons entre eux seront rares. Par conséquent, des gènes voisins chez des enfants de robots tout rouge et tout marron impliquent plus fréquemment le même phénotype (tous les deux rouges ou tous les deux marrons) ; les biologistes peuvent donc déduire l'ordre en considérant la fréquence des phénotypes chez les paires. À l'aide de tels arguments, Sturtevant a construit la première carte génétique pour six gènes chez les drosophiles en 1913.

Bien que la génétique humaine soit plus compliquée que celle des robots, le modèle simpliste du robot capte de nombreuses idées informatiques cachées derrière les algorithmes de cartographie génétique. L'une des complications est que les gènes humains vont par paires (sans dire qu'ils sont répartis sur 23 chromosomes). Dans chaque paire, un gène est hérité de la mère et l'autre du père. Par conséquent, le génome humain peut contenir un gène à l'état 1 (œil rouge) sur un chromosome et un gène à l'état 0 (œil marron) sur l'autre chromosome de la même paire. Si $P_1 \dots P_n | \mathcal{P}_1 \dots \mathcal{P}_n$ représente le génome du père (chaque gène est présent dans deux copies P_i et $\mathcal{P}_i$) et $M_1 \dots M_n | \mathcal{M}_1 \dots \mathcal{M}_n$ celui de la mère, alors le génome d'un enfant est représenté par $p_1 \dots p_n | m_1 \dots m_n$, où p_i vaut P_i ou $\mathcal{P}_i$ et m_i est égal à M_i ou $\mathcal{M}_i$. Par exemple, le père 11|00 et la mère 00|00 peuvent avoir quatre types d'enfants différents : 11|00 (pas de recombinaison), 10|00 (recombinaison), 01|00 (recombinaison) et 00|00 (pas de recombinaison). Les idées fondamentales cachées derrière la cartographie génétique de l'être humain et du robot sont semblables : comme la recombinaison entre des gènes proches est rare, la proportion de recombinants parmi les enfants donne une indication de la distance entre les gènes le long du chromosome.

Une autre complication provient de ce que les différences dans les génotypes n'induisent pas toujours des différences dans les phénotypes. Par exemple, les humains possèdent un gène appelé *groupe sanguin* qui a trois états — A, B et O — dans la population humaine. Il existe six génotypes possibles pour ce gène — AA, AB, AO, BB, BO et OO — mais seulement quatre phénotypes. Dans ce cas, le phénotype ne nous permet pas de déterminer le génotype sans ambiguïté. De ce point de vue, la couleur des yeux ou le groupe sanguin ne sont peut-être pas les meilleures caractéristiques à utiliser pour construire les cartes génétiques. Les biologistes proposent d'utiliser des *marqueurs génétiques* comme un substitut convenable pour les gènes en cartographie génétique. Pour cartographier un nouveau gène, il est nécessaire d'avoir un grand nombre de marqueurs connus qui, idéalement, devraient se situer à intervalles réguliers le long des chromosomes.

Notre capacité à cartographier les gènes chez les robots est fondée sur la variabilité des phénotypes chez différents robots. Par exemple, si tous les robots avaient les yeux marrons, le gène des yeux serait impossible à repérer. Il y a de nombreuses variations dans le génome humain, qui ne sont pas directement exprimées dans les phénotypes. Par exemple, si la moitié des humains avait le nucléotide A à une certaine position dans le génome et que l'autre moitié

avait le nucléotide T à la même position, ce serait un bon marqueur pour la cartographie génétique. Une telle mutation peut se produire à l'extérieur de tout gène et ne pas affecter le phénotype du tout. Botstein *et al.*, 1980 [44] ont proposé d'utiliser de telles positions variables comme des marqueurs génétiques pour la cartographie. Comme il est expérimentalement impossible de prélever des lettres à une position donnée du génome, ils ont suggéré une technique appelée *polymorphisme de longueur des fragments de restriction* (RFLP pour *restriction fragment length polymorphism*) pour étudier la variabilité.

Hamilton Smith découvrit en 1970 que l'*enzyme de restriction Hin*dII coupe les molécules d'ADN à chaque occurrence de la séquence GTGCAC ou GTTAAC (sites de restriction). En analyse RFLP, l'ADN humain est coupé par une enzyme de restriction comme *Hin*dII à chaque occurrence du site de restriction en un million de fragments de restriction environ, mesurant chacun quelques milliers de nucléotides de long. Cependant, toute mutation qui affecte l'un des sites de restriction (GTGCAC ou GTTAAC pour *Hin*dII) met hors service l'une des coupures et fusionne deux fragments de restriction A et B séparés par ce site en un unique fragment $A + B$. Le point crucial de l'analyse RFLP est la détection du changement dans la longueur des fragments de restriction.

L'électrophorèse sur gel sépare les fragments de restriction et on utilise une sonde d'ADN marquée pour déterminer la taille du fragment de restriction qui s'hybride à cette sonde. La variabilité dans la longueur de ces fragments de restriction chez différents individus sert de marqueur génétique, car la mutation d'un seul nucléotide peut détruire (ou créer) le site pour une enzyme de restriction et modifier la longueur du fragment correspondant. Par exemple, si une sonde d'ADN marquée s'hybride à un fragment A et si un site de restriction séparant les fragments A et B est détruit par une mutation, alors la sonde détecte $A + B$ au lieu de A. Kan et Dozy, 1978 [183] ont trouvé un nouveau diagnostic pour la drépanocytose en identifiant un marqueur RFLP situé près du gène de la drépanocytose.

L'analyse RFLP a transformé la cartographie génétique en une course hautement compétitive et les réussites se sont succédées de façon très rapprochée, avec la découverte des gènes responsables de la maladie de Huntington (Gusella *et al.*, 1983 [143]), de la dystrophie musculaire de Duchenne (Davies *et al.*, 1983 [81]) et du rétinoblastome (Cavenee *et al.*, 1985 [60]). Dans une publication référence, Donis-Keller *et al.*, 1987 [88] ont construit la première carte RFLP du génome humain, en positionnant un marqueur RFLP pour approximativement dix millions de nucléotides. Dans cette étude, 393 sondes aléatoires ont été utilisées pour étudier le RFLP dans 21 familles sur trois générations. Finalement, une analyse informatique des recombinaisons permit d'ordonner les marqueurs RFLP sur les chromosomes.

En 1985, les études de recombinaison restreignaient la recherche pour le gène de la mucoviscidose à un secteur du chromosome 7 situé entre les marqueurs *met* (un gène impliqué dans le cancer) et D7S8 (un marqueur RFLP). La longueur de ce secteur était d'environ un million de nucléotides et il s'est écoulé un certain

temps avant que le gène de la mucoviscidose ne soit découvert. La cartographie génétique fait place à la cartographie physique pour limiter davantage encore la recherche.

1.3 Cartographie physique

La cartographie physique peut être comprise grâce à l'analogie suivante. Imaginons quelques exemplaires d'un livre que l'on découpe avec des ciseaux en plusieurs milliers de morceaux. Chaque exemplaire est coupé d'une façon qui lui est propre, si bien qu'un morceau d'un exemplaire puisse recouvrir un morceau d'un autre. Pour chaque morceau et chaque mot d'une liste de mots-clés, on sait si le morceau contient le mot. Ces données étant connues, nous voulons déterminer la maquette des recouvrements des morceaux.

Le procédé commence en cassant la molécule d'ADN en petits morceaux (avec des enzymes de restriction, par exemple) ; dans le projet pour la mucoviscidose, l'ADN était cassé en morceaux d'environ 50 kb de long. Pour étudier chaque morceau individuellement, les biologistes ont besoin d'obtenir plusieurs copies de chacun d'eux. Ceci est réalisé en *clonant* les morceaux. Le clonage incorpore un fragment d'ADN dans un hôte auto-réplicant. Le procédé d'auto-réplication crée alors un grand nombre de copies de ce fragment, permettant ainsi d'étudier sa structure. Un fragment reproduit de cette façon est appelé un *clone*.

Par suite, les biologistes obtiennent une *banque de clones* constituée de milliers de clones (chacun représentant un court fragment d'ADN) de la même molécule d'ADN. Les clones de la banque peuvent se chevaucher (ceci peut être accompli en coupant l'ADN avec des enzymes distinctes produisant des fragments de restriction qui se chevauchent). Après avoir construit une banque de clones, les biologistes veulent *ordonner* ces derniers, c'est-à-dire reconstruire les positions relatives des clones le long de la molécule d'ADN. Cette information est perdue lors de la construction de la banque de clones et la reconstruction commence avec la *prise de l'empreinte digitale* (ou *fingerprint*) des clones. L'idée est de décrire chaque clone à l'aide d'une empreinte digitale que l'on détermine facilement et qui peut être vue comme un ensemble de « mots-clés » pour le clone. Si deux clones ont des chevauchements importants, leurs empreintes digitales se ressembleront. S'il est peu probable que des clones qui ne se chevauchent pas aient des empreintes similaires, alors cela permet à un biologiste de faire la distinction entre des clones qui se chevauchent et d'autres qui ne se chevauchent pas et ainsi de reconstruire l'ordonnancement des clones (cartographie physique). Les tailles des fragments ou les listes de sondes qui s'hybrident à un clone fournissent de telles empreintes digitales.

Pour cartographier le gène de la mucoviscidose, les biologistes ont utilisé des techniques de cartographie physique appelées *marche le long d'un chromosome* et *saut chromosomique*. Rappelons que le gène de la mucoviscidose était lié au RFLP D7S8. La sonde correspondant à ce RFLP peut être utilisée pour

trouver un clone qui le contient. Ce clone peut être séquencé et l'une de ses extrémités utilisée pour créer une nouvelle sonde encore plus proche du gène de la mucoviscidose. Ces sondes peuvent être utilisées pour trouver de nouveaux clones et pour *marcher* depuis le D7S8 au gène de la mucoviscidose. Après de multiples itérations, on peut séquencer des centaines de kilobases d'ADN à partir d'une région entourant le gène marqueur. Si celui-ci est étroitement lié au gène qui nous intéresse, alors ce gène peut éventuellement être séquencé, lui aussi. Dans le projet pour la mucovisidose, une longueur totale de 249 kb a été clonée en 58 fragments d'ADN.

Les projets de marche le long d'un chromosome sont plutôt complexes et fastidieux. L'un des obstacles est que toutes les régions de l'ADN ne seront pas présentes dans la banque de clones, puisque certaines régions génomiques ont tendance à être instables lorsqu'elles sont clonées chez les bactéries. Collins *et al.*, 1987 [73] ont développé le *saut chromosomique*, qui a été utilisé avec succès pour cartographier le secteur contenant le gène de la mucoviscidose.

Bien que leurs concepts soient attrayants, la marche le long d'un chromosome et le saut chromosomique sont trop laborieux pour cartographier des génomes entiers ; ils sont faits pour cartographier des gènes individuellement. Une carte pré-construite couvrant le génome complet économiserait beaucoup d'efforts pour cartographier *tout* nouveau gène.

Des empreintes digitales différentes mènent à des problèmes de cartographie différents. Dans le cas d'empreintes basées sur l'hybridation avec des sondes courtes, une sonde peut s'hybrider à de nombreux clones. Pour le problème d'assemblage de cartes avec n clones et m sondes, les données d'hybridation sont constituées d'une matrice (d_{ij}) de dimensions $n \times m$, où d_{ij} vaut 1 si le clone C_i contient la sonde p_j et 0 sinon (voir figure 1.1). Notons que ces renseignements n'indiquent pas le nombre de fois qu'une sonde apparaît dans un clone donné et qu'ils ne donnent pas non plus l'ordre des sondes dans un clone.

L'approximation la plus simple de cartographie physique est le problème de la plus courte chaîne couvrante. Soit S une chaîne dans l'alphabet des sondes $p_1, \ldots, p_m$. Une chaîne S *recouvre* un clone C s'il existe une sous-chaîne de S contenant exactement le même ensemble de sondes que C (on ne tient compte ni de l'ordre ni des multiplicités des sondes dans la sous-chaîne). Une chaîne de la figure 1.1 recouvre chacun des neuf clones correspondant aux données d'hybridation.

Problème de la plus courte chaîne couvrante À partir des données d'hybridation, trouver une des chaînes les plus courtes dans l'alphabet des sondes, qui recouvre tous les clones.

Avant d'utiliser des sondes pour la cartographie de l'ADN, les biologistes construisaient des cartes de restriction de clones et les utilisaient comme empreintes pour ordonner les clones. La *carte de restriction* d'un clone est une liste ordonnée de fragments de restriction. Si deux clones ont des cartes de res-

triction qui ont quelques fragments consécutifs en commun, il y a des chances pour qu'ils se chevauchent. Avec cette stratégie, Kohara *et al.*, 1987 [204] ont assemblé une carte de restriction du génome *E. Coli* avec cinq millions de paires de bases.

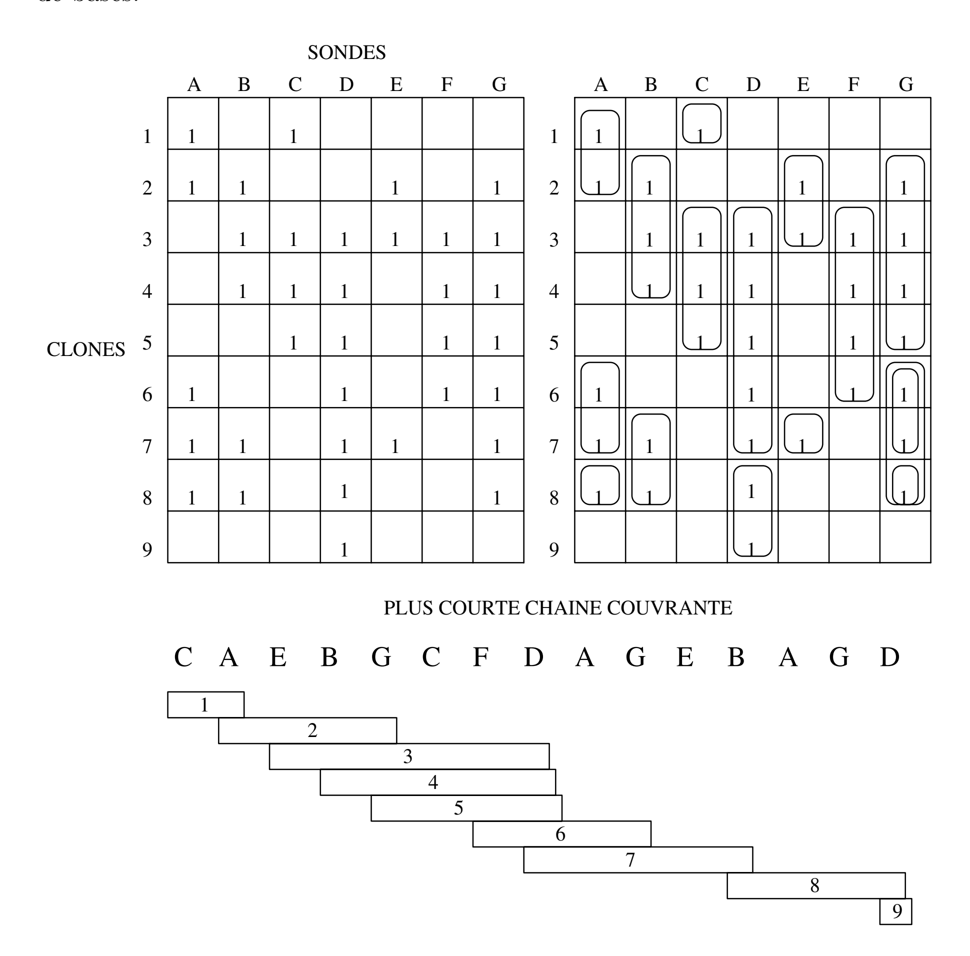


Figure 1.1 – Données d'hybridation et plus courte chaîne couvrante.

Pour construire une carte de restriction d'un clone, les biologistes utilisent différentes techniques biochimiques pour obtenir une information indirecte concernant la carte, ainsi que des méthodes combinatoires pour reconstruire la carte à partir de ces données. Le problème peut souvent être formulé comme un recouvrement de positions de points, où l'on ne connaît que quelques distances entre des paires de points.

De nombreuses techniques de cartographie mènent au problème combinatoire suivant. Si X est un ensemble de points alignés, alors ΔX représente

le multi-ensemble de toutes les distances entre des couples de points de X : $\Delta X = \{|x_1 - x_2| : x_1, x_2 \in X\}$. En cartographie de restriction, on donne un sous-ensemble $E \subset \Delta X$ correspondant aux données expérimentales sur les longueurs des fragments et le problème consiste à reconstruire X à partir de la seule connaissance de E. Dans le *problème de la digestion partielle (PDP)*, les expériences fournissent des données concernant *toutes* les distances entre les paires de sites de restriction et on a $E = \Delta X$.

Problème de la digestion partielle Étant donné l'ensemble ΔX, reconstruire X.

Ce problème est aussi connu sous le nom du problème de l'*autoroute* en informatique. Supposons que l'on connaisse l'ensemble formé de toutes les distances entre chaque couple de sorties d'une autoroute. Est-il possible de reconstruire la « géographie » de celle-ci à partir de ces données, c'est-à-dire trouver les distances entre le début de l'autoroute et chaque sortie ? Si, à la place des sorties d'autoroute, on considère le clivage des sites d'ADN par une enzyme de restriction et si l'on parvient à digérer l'ADN de façon que les fragments formés par *toutes* les paires de coupures soient présents dans la digestion, alors les tailles des fragments d'ADN qui en résultent correspondent aux distances entre les sorties d'autoroute.

On ne connaît pas encore d'algorithme polynomial pour résoudre cette énigme apparemment triviale.

1.4 Séquençage

Imaginez plusieurs exemplaires d'un livre que l'on coupe avec des ciseaux en dix millions de petits morceaux. Chaque copie est coupée d'une façon qui lui est propre, si bien qu'un morceau d'un exemplaire peut recouvrir un morceau d'un autre. En supposant que l'on ait perdu un million de pièces et que l'on ait éclaboussé les neuf millions restantes avec de l'encre, vous devez essayer de retrouver le texte original. Après avoir fait cela, vous aurez un aperçu de ce qu'est un problème de séquençage de l'ADN. La technologie de séquençage classique permet à un biologiste de lire de courts fragments (de 300 à 500 lettres) lors d'expériences (chacun de ces fragments correspond à une pièce parmi les dix millions). Les bio-informaticiens doivent assembler le génome entier à partir de ces courts fragments, une tâche pas très différente de l'assemblage d'un livre à partir de millions de morceaux de papier. Le problème est compliqué par des erreurs expérimentales inévitables (taches d'encre).

L'approximation la plus simple (bien qu'un peu naïve) du séquençage de l'ADN correspond au problème suivant :

Problème de la super-chaîne la plus courte Étant donné un ensemble de chaînes $s_1, \ldots, s_n$, trouver la chaîne la plus courte s telle que chaque s_i apparaisse comme une sous-chaîne de s.

La figure 1.2 présente deux super-chaînes pour l'ensemble des huit chaînes de trois lettres dans l'alphabet 0-1. La première super-chaîne (triviale) est obtenue par concaténation de ces huit chaînes, tandis que la seconde est une super-chaîne de longueur minimale. Celle-ci est liée à la solution du problème « du cambrioleur intelligent et de la combinaison du coffre » (le nombre minimal de tests qu'un cambrioleur doit réaliser pour essayer tous les codes possibles à k lettres).

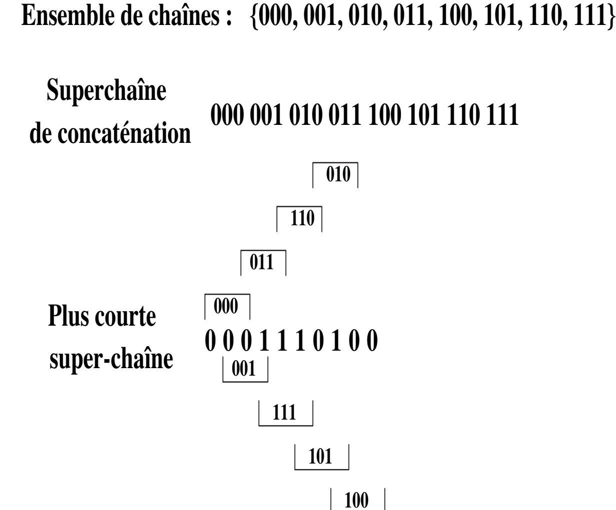


Figure 1.2 – Super-chaînes pour l'ensemble des huit chaînes de trois lettres dans l'alphabet 0-1.

Comme le problème de la super-chaîne la plus courte est NP-complet, un certain nombre d'heuristiques ont été proposées. Les premiers algorithmes de séquençage de l'ADN utilisaient une simple stratégie *gloutonne* : fusionner de façon répétée une paire de chaînes ayant un chevauchement maximal, jusqu'à ce qu'il ne reste plus qu'une seule chaîne.

Bien que le séquençage conventionnel de l'ADN soit désormais une procédure rapide et efficace, il nécessitait beaucoup de temps et était difficile à automatiser il y a encore dix ans. En 1988, quatre groupes de biologistes proposèrent indépendemment et simultanément une nouvelle approche appelée séquençage par hybridation (SBH). Ils suggérèrent de construire une *puce à ADN (matrice)* miniature contenant des milliers de courts fragments d'ADN travaillant

comme la mémoire d'une puce. Chacun de ces courts fragments révèle de l'information sur un fragment d'ADN inconnu et tous ces bouts d'information, une fois combinés, sont censés résoudre l'énigme du séquençage de l'ADN. En 1988, quasiment personne ne croyait que cette idée allait fonctionner ; les problèmes biochimiques (synthétiser des milliers de courts fragments d'ADN sur la surface de la puce) et les problèmes combinatoires (reconstruire la séquence à partir des résultats de la matrice) semblaient trop compliqués. Désormais, la construction de puces à ADN avec des milliers de sondes est devenue une industrie.

Étant donné un fragment d'ADN avec une séquence inconnue de nucléotides, une puce à ADN fournit sa composition en l-uplets, c'est-à-dire l'information sur toutes les sous-chaînes de longueur l contenues dans ce fragment (les positions de ces sous-chaînes sont inconnues).

Problème du séquençage par hybridation Reconstruire une chaîne à partir de sa composition en l-uplets.

Bien que les puces à ADN aient été inventées à l'origine pour le séquençage de l'ADN, très peu de fragments ont été séquencés avec cette technique (Drmanac *et al.*, 1993 [90]). Le problème est que l'infidélité du processus d'hybridation amène des erreurs dans la décomposition en l-uplets. Comme cela arrive souvent en biologie, les puces à ADN furent un succès, non pas dans le cadre du problème pour lequel elles avaient été inventées initialement, mais dans différentes applications de génomique fonctionnelle et pour la détection de mutations.

Bien que le séquençage conventionnel de l'ADN et le SBH soient des approches très différentes, les problèmes informatiques correspondants sont similaires. En fait, le SBH est un cas particulier du problème de la super-chaîne la plus courte, lorsque toutes les chaînes $s_1, \ldots, s_n$ représentent l'ensemble de toutes les sous-chaînes de taille fixée de s. Cependant, contrairement au cas du séquençage, il existe un algorithme linéaire simple pour le problème du SBH.

1.5 Recherche de similitudes

Après le séquençage, les biologistes n'ont habituellement aucune idée de l'utilité des gènes trouvés. En espérant découvrir un indice sur leurs fonctions, ils tentent de trouver des similitudes entre des gènes nouvellement séquencés et d'autres déjà séquencés dont ils connaissent les fonctions. Voici un exemple frappant de découverte biologique faite lors d'une recherche de similitudes qui se déroula en 1984, alors que des scientifiques utilisaient une technique informatique simple pour comparer l'oncogène ν-*sys* (cause de cancer) nouvellement découvert à tous les gènes connus. À leur grand étonnement, celui-ci correspondait à un gène normal impliqué dans la croissance et le développement. Il devint soudain clair que le cancer pouvait être causé par un gène de croissance

normal qui se met en marche au mauvais moment (Doolittle *et al.*, 1983 [89] et Waterfield *et al.*, 1983 [353]).

En 1879, Lewis Carroll proposait aux lecteurs de *Vanity Fair* le jeu suivant : transformer un mot anglais en un autre mot en passant par une série de mots intermédiaires, dans laquelle chaque mot ne diffère du suivant que d'une seule lettre. Pour transformer *head* en *tail*, on n'a besoin que de quatre intermédiaires : *head* → *heal* → *teal* → *tell* → *tall* → *tail*. Levenshtein, 1966 [219] a introduit la notion de *distance d'édition* entre deux chaînes : il s'agit du nombre minimal d'opérations élémentaires nécessaires pour passer d'une chaîne à l'autre. Les opérations élémentaires sont l'insertion d'un symbole, la suppression d'un symbole et la substitution d'un symbole à un autre. La plupart des algorithmes de comparaison de séquences sont liés au calcul de la distance d'édition, avec cet ensemble d'opérations élémentaires ou un ensemble légèrement différent.

Comme une mutation dans l'ADN représente un processus naturel d'évolution, la distance d'édition est une mesure naturelle de la similitude entre des fragments d'ADN. La similitude entre des séquences d'ADN peut être l'indice d'une origine évolutive commune (comme la similitude entre les gènes de la globine chez les humains et les chimpanzés) ou d'une fonction commune (comme la similitude entre l'oncogène *ν-sys* et une hormone de stimulation de la croissance).

Si les opérations d'édition se limitent aux insertions et aux suppressions (sans substitution), alors le problème de la distance d'édition est équivalent au problème de *la plus longue sous-séquence commune* (LCS pour *longest common subsequence*). Étant données deux chaînes $V = v_1 \dots v_n$ et $W = w_1 \dots w_m$, une *sous-séquence commune* à V et W de longueur k est une séquence d'indices $1 \leq i_1 < \dots < i_k \leq n$ et $1 \leq j_1 < \dots < j_k \leq m$ telle que

$$v_{i_t} = w_{j_t} \text{ pour } 1 \leq t \leq k$$

Notons $LCS(V, W)$ la longueur d'une des *plus longues sous-séquences communes* (LCS) à V et W. Par exemple, on a : *LCS* (A**TCT**G**A**T, **T**G**C**A**TA**)= 4 (les lettres formant la LCS sont en gras). Il est clair que $n + m - 2LCS(V, W)$ est le nombre minimal d'insertions et de suppressions nécessaires pour transformer V en W.

Problème de la plus longue sous-séquence commune Étant données deux chaînes, trouver leurs plus longues sous-séquences communes.

Lorsque la zone située autour du gène de la mucoviscidose a été séquencée, les biologistes l'ont comparé aux gènes connus et ont trouvé quelques similitudes avec un fragment d'environ 6500 nucléotides de long appelé *protéine de liaison ATP* qui avait déjà été découvert. Ces protéines franchissent plusieurs fois la membrane cellulaire et fonctionnent comme des conduits pour le transport des ions au travers de cette membrane. Ceci semblait être une fonction plausible pour un gène de la mucoviscidose, étant donné que cette maladie implique des

sécrétions anormales. La similitude fit également ressortir deux sites de liaison ATP conservés (les protéines ATP fournissent de l'énergie pour de nombreuses réactions dans la cellule) et éclaira le mécanisme endommagé dans les gènes défectueux de la mucoviscidose. Par suite, le gène de la mucoviscidose a été appelé *régulateur de la conductance transmembranaire*.

1.6 Prédiction génétique

Connaître la place approximative d'un gène ne mène pas encore au gène lui-même. Par exemple, le gène de la maladie de Huntington a été cartographié en 1983, mais il est resté insaisissable jusqu'en 1993. En revanche, le gène de la mucoviscidose a été cartographié en 1985 et découvert en 1989.

Dans les formes de vie primitives comme les bactéries, les gènes figurent dans l'ADN sous forme de chaînes continues. Chez les êtres humains (et autres mammifères), la situation est beaucoup moins simple. Un gène humain, constitué d'environ 2000 lettres, est typiquement cassé en deux sous-fragments appelés *exons*. Ces exons peuvent être mêlés, de façon apparemment aléatoire, en une section d'ADN chromosomique allant jusqu'à un million de lettres. Un gène humain typique peut contenir dix exons ou plus. Le gène *BRCA1*, impliqué dans le cancer du sein, comporte vingt-sept exons.

Cette situation peut être comparée à un article de magazine qui commencerait en page 1, continuerait en page 13, puis reprendrait aux pages 43, 51, 53, 74, 80 et 91, avec des pages de publicité et d'autres articles entre deux. On ne comprend pas pourquoi ces sauts se produisent, on ne sait pas quel en est le but. 97 % du génome humain est constitué de publicité ou de ce que l'on appelle de l'ADN « poubelle ».

Les sauts sont incompatibles d'une espèce à l'autre. Un « article » paru dans l'édition d'un magazine génétique d'un insecte sera imprimé différemment du même article paru dans l'édition d'un ver. La pagination sera complètement différente : l'information qui apparaît sur une seule page dans l'édition humaine peut être cassée en deux dans la version du blé, ou vice versa. Les gènes eux-mêmes, bien que liés, sont assez différents. Le gène de l'édition de la souris est écrit dans le langage de la souris, le gène de l'édition humaine en langage humain. C'est un peu comme l'allemand et l'anglais : de nombreux mots sont similaires, mais ce n'est pas le cas de beaucoup d'autres.

Prédire un nouveau gène dans une séquence d'ADN nouvellement séquencée est un problème difficile. De nombreuses méthodes permettant de faire la distinction entre la publicité et l'histoire véritable dépendent des statistiques. Pour continuer la comparaison avec un magazine, c'est un peu comme si l'on se lisait le magazine à l'envers et que l'on trouvait que les « histoires » de gènes humains ont moins de chance de contenir des phrases comme « à vendre », des numéros de téléphone ou encore le symbole de l'euro. En revanche, une approche combinatoire de prédiction génétique utilise au préalable des gènes séquencés comme gabarit pour permettre de reconnaître des gènes nouvellement

séquencés. Au lieu d'employer les propriétés statistiques des exons, cette méthode tente de résoudre l'énigme combinatoire suivante : trouver un ensemble de blocs (exons candidats) dans une séquence génomique, dont la concaténation (le collage) s'ajuste à l'une des protéines connues. La figure 1.3 illustre ce jeu pour la séquence « génomique » suivante :

'twas brilliant thrilling morning and the slimy hellish lithe doves
gyrated and gambled nimbly in the waves

dont les différents blocs « composent » la célèbre « protéine cible » de Lewis Carroll :

't was brillig, and the slithy toves did gyre and gimble in the wabe

'T WAS BRILLIG, AND THE SLITHY TOVES DID GYRE AND GIMBLE IN THE WABE

T WAS BRILLI	G, AND THE SL	THE DOVES GYRATED AND GAMBLED	IN THE WAVE
T WAS BRILLI	G, AND THE SL	THE DOVES GYRATED	NIMBLY IN THE WAVE
T HRILLING	AND	HEL LISH	DOVES GYRATED AND GAMBLED	IN THE WAVE
T HRILLING	AND	HEL LISH	DOVES GYRATED	NIMBLY IN THE WAVE



Figure 1.3 – Problème de l'alignement du collage : les assemblements de blocs s'ajustant le mieux à la « protéine cible » de Lewis Carroll.

Cette énigme combinatoire aboutit au problème suivant :

Problème d'alignement du collage Soient G une chaîne appelée *séquence génomique*, T une chaîne appelée *séquence cible* et $\mathcal{B}$ un ensemble de sous-chaînes de G. Étant données G, T et $\mathcal{B}$, trouver un ensemble de chaînes de $\mathcal{B}$ qui ne se chevauchent pas et dont la concaténation s'ajuste au mieux à la séquence cible (autrement dit, la distance d'édition entre la concaténation de ces chaînes et la cible est minimale parmi tous les ensembles de blocs de $\mathcal{B}$).

1.7 Analyse de mutations

L'un des défis de la recherche génétique est de savoir quand le gène recherché a été séquencé, étant donné qu'on ne connaît rien sur la structure de ce gène. Dans le cas de la mucoviscidose, la prédiction génétique et la similitude de séquences ont fourni quelques indices concernant le gène, mais elles n'ont pas écarté les autres gènes candidats. En particulier, trois autres fragments étaient suspects. Si un gène suspect était vraiment un gène malade, les personnes atteintes auraient des mutations à ce niveau. Tout gène de ce type sera sujet à un nouveau séquençage chez de nombreux individus pour vérifier cette hypothèse. On a trouvé dans le gène de la mucoviscidose une mutation (suppression de trois nucléotides, provoquant la suppression d'un acide aminé) commune à des personnes atteintes. C'était un fil conducteur et on a créé des amorces PCR pour passer au crible un grand nombre d'individus pour cette mutation. Celle-ci a été trouvée chez 70% des patients atteints de mucoviscidose, prouvant ainsi qu'elle était à l'origine de la maladie. Des centaines de mutations diverses contiennent les 30% de gènes supplémentaires responsables de la mucoviscidose, rendant la maladie difficile à diagnostiquer d'un point de vue médical. Des puces à ADN consacrées à la mucoviscidose pourraient être très efficaces pour passer au crible les populations pour cette mutation.

La recherche de similitudes, la reconnaissance génétique et l'analyse de mutations font naître un certain nombre de problèmes statistiques. Si deux séquences sont semblables à 45%, est-il probable qu'elles soient authentiquement liées, ou est-ce juste une question de hasard ? Dans les fragments d'ADN, on trouve souvent des gènes ayant une haute fréquence de nucléotides CG (*îlots CG*). Le gène de la mucoviscidose, en particulier, est situé à l'intérieur d'un îlot CG. Quel niveau de contenu CG constitue une indication d'un îlot CG et quand s'agit-il juste de hasard ? Voici quelques exemples de problèmes statistiques correspondants :

Problème de la longueur attendue de la LCS Trouver la longueur attendue de la LCS pour deux chaînes aléatoires de longueur n.

Problème statistique des chaînes Trouver l'espérance et la variance du nombre d'occurrences d'une chaîne donnée dans un texte aléatoire.

1.8 Comparaison génomique

Comme nous l'avons vu avec la mucoviscidose, la recherche de gènes humains peut être une entreprise longue et laborieuse. Souvent, les études génétiques portant sur des désordres génétiques similaires chez les animaux peut accélérer le processus.

Le syndrome de Waardenburg est un désordre génétique héréditaire qui se traduit par une perte auditive et une dysplasie pigmentaire. La cartographie génétique a restreint la recherche pour le gène du syndrome de Waardenburg au chromosome 2 humain, mais sa position exacte reste inconnue. Il existait un autre indice qui attira l'attention sur le chromosome 2. Depuis longtemps, une équipe de biologistes a étudié des souris sous l'angle de la mutation et l'une d'elles, nommée *tachetée*, présentait des taches blanches, une maladie aux manifestations semblables au syndrome de Waardenburg. Par le biais de sélections (qui sont plus faciles à faire chez les souris que chez les êtres humains), le gène *tache* a été cartographié sur le chromosome 2 de la souris. Alors que la cartographie du gène se poursuivait, il devint clair qu'il existait des groupes de gènes étroitement liés les uns aux autres chez les deux espèces. Le brassage du génome durant l'évolution n'est pas complet ; des blocs de matériel génétique restent intacts, même s'il se produit de multiples réarrangements chromosomiques. Par exemple, le chromosome 2 chez les humains est construit à partir de fragments qui sont semblables aux fragments de l'ADN de la souris se situant sur les chromosomes 1, 2, 6, 8, 11, 12 et 17 (voir figure 1.4). Par conséquent, cartographier un gène chez les souris donne souvent un indice sur la position d'un gène humain qui lui est apparenté.

Malgré quelques différences au niveau de l'apparence et des coutumes, les hommes et les souris sont génétiquement très semblables. Dans un article novateur, Nadeau et Taylor, 1984 [248] ont estimé que, de façon assez surprenante, peu de réarrangements génomiques (178 ± 39) se sont produits depuis la divergence entre l'être humain et la souris, il y a 80 millions d'années. Les génomes de la souris et de l'humain peuvent être vus comme une collection d'environ 200 fragments qui sont brassés (réarrangés) chez les souris, comme chez les humains. Si le gène d'une souris est cartographié dans l'un de ces fragments, alors le gène humain correspondant sera situé dans un fragment chromosomique en rapport avec ce gène de la souris. Étant donnée la position d'un gène souris, une carte génétique comparative homme-souris indique la position du gène humain correspondant.

Les réarrangements génomiques sont une anomalie chromosomique plutôt courante ; ils sont associés à des maladies génétiques comme le syndrome de Down. Les réarrangements génomiques sont souvent asymptomatiques : on estime que $0,2\%$ de la population porte un réarrangement chromosomique asymptomatique.

Dobzhansky et Sturtevant, 1938 [87] ont été les premiers à utiliser l'analyse des réarrangements génomiques en biologie moléculaire ; ils ont publié un article important qui présente un scénario de réarrangement avec 17 inversions pour les espèces de mouches à fruit (*drosophiles*). Dans la forme la plus simple, les réarrangements peuvent être modélisés comme un problème combinatoire consistant à trouver les plus courtes séries d'*inversions* pour transformer un génome en un autre génome. L'ordre des gènes dans un organisme est représenté par une permutation $\pi = \pi_1 \pi_2 \dots \pi_n$. Une *inversion* $\rho(i,j)$ a pour effet d'inverser l'ordre des gènes $\pi_i \pi_{i+1} \dots \pi_j$ et de transformer $\pi = \pi_1 \dots \pi_{i-1} \pi_i \dots \pi_j \pi_{j+1} \dots \pi_n$ en



Figure 1.4 – Carte physique comparative homme-souris.

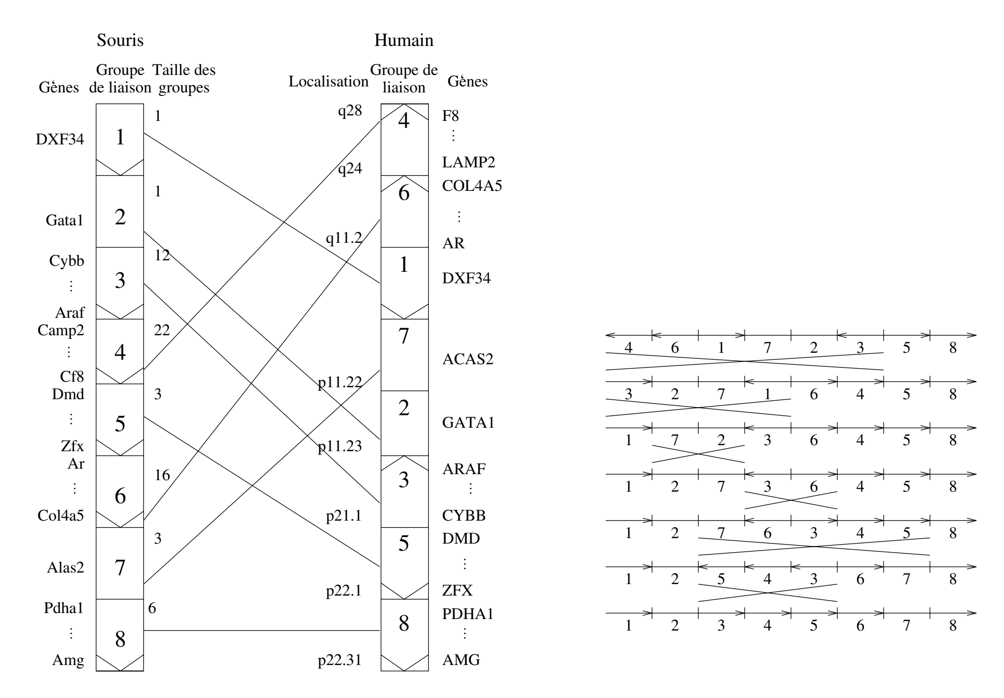


Figure 1.5 – « Transformation » d'un chromosome X humain en un chromosome X de souris.

$\pi \cdot \rho(i,j) = \pi_1 \dots \pi_{i-1}\pi_j\pi_{j-1} \dots \pi_{i+1}\pi_i\pi_{j+1} \dots \pi_n$. La figure 1.5 présente un *scénario de réarrangement* qui décrit la transformation d'un chromosome X humain en un chromosome X de souris.

Problème de la distance d'inversion Étant données des permutations π et σ, trouver une suite d'inversions $\rho_1, \rho_2, \dots, \rho_t$ vérifiant $\pi \cdot \rho_1 \cdot \rho_2 \cdots \rho_t = \sigma$ telle que t soit minimal.

1.9 Protéomique

Dans de nombreux organismes vivants, des cellules meurent à des moments particuliers ; ce processus est appelé la *mort cellulaire programmée*. La mort peut être le résultat d'un manque en vue d'acquérir des facteurs de survie et peut être amorcée par l'expression de certains gènes. Par exemple, dans un nématode en développement, la mort de cellules particulières dans le système nerveux peut être empêchée par des mutations dans plusieurs gènes dont la fonction est à l'étude. Cependant, les approches fondées sur l'ADN décrites précédemment ne sont pas bien adaptées pour trouver des gènes impliqués dans la mort cellulaire programmée.

Le mécanisme de mort cellulaire est un système complexe composé de nombreux gènes. Alors que beaucoup de protéines correspondant à ces gènes candidats ont été identifiées, leurs rôles et la façon dont elles interviennent dans la mort cellulaire programmée sont encore mal compris. La difficulté est que l'ADN de ces gènes candidats est difficile à isoler, plus difficile en tout cas que les protéines correspondantes. Cependant, il n'existe pas encore de méthode sûre pour le séquençage protéique et la séquence de ces gènes candidats était toujours inconnue il y a peu.

Récemment, une nouvelle approche du séquençage protéique par spectrométrie de masse est apparue, permettant le séquençage de nombreuses protéines impliquées dans la mort cellulaire programmée. En 1996, le séquençage protéique aboutit à l'identification de la protéine *FLICE*, qui est en jeu dans le complexe de signalisation de mort induite (Muzio *et al.*, 1996 [244]). Dans ce cas, la recherche génétique a débuté par le séquençage d'une protéine (plutôt que de l'ADN) et a mené par la suite au clonage de ce gène *FLICE*. L'exceptionnelle sensibilité de la spectrométrie de masse a ouvert de nouvelles perspectives expérimentales et informatiques pour le séquençage protéique et a fait de cette technique une méthode de choix dans de nombreux domaines.

Le séquençage protéique a, depuis longtemps, fasciné les spécialistes de la spectrométrie de masse (Johnson et Biemann, 1989 [182]). Cependant, il a fallu attendre ces dernières années, avec le développement des systèmes automatisés et les algorithmes récemment mis au point, pour que le séquençage protéique à haut débit devienne une réalité, voire une porte ouverte au « séquençage protéomique ». Actuellement, la plupart des protéines sont identifiées par la recherche dans des bases de données (Eng *et al.*, 1994 [97] et Mann et Wilm, 1994 [230]), qui repose sur la capacité à « chercher la réponse au dos du livre ». Bien que la recherche dans des bases de données soit très utile pour les génomes déjà séquencés, un biologiste qui tente de trouver un *nouveau* gène a besoin d'algorithmes récemment mis au point plutôt que d'algorithmes de recherche dans des bases de données.

En quelques secondes, un spectromètre de masse est capable de casser un peptide en morceaux (des *ions*) et de mesurer leur *masse*. L'ensemble des masses qui en résulte forme le *spectre* d'un peptide. Le *problème du séquençage peptidique* consiste à reconstruire un peptide à partir de son spectre. Pour un procédé « idéal » de fragmentation et un spectromètre de masse « idéal », le problème du séquençage peptidique est simple. Dans la pratique, le séquençage *de novo* d'un peptide demeure un problème ouvert, car les spectres sont difficiles à interpréter.

Sous sa forme la plus simple, le séquençage protéique par spectrométrie de masse correspond au problème suivant. Soit A l'ensemble des acides aminés ; on note $m(a)$ la masse moléculaire d'un acide $a \in A$. Un *peptide* (parent) $P = p_1, \ldots, p_n$ est une séquence d'acides aminés de masse $m(P) = \sum m(p_i)$. Un *peptide partiel* $P' \subset P$ est une sous-chaîne $p_i \ldots p_j$ de P de masse $\sum_{i \leq t \leq j} m(p_t)$. Le *spectre théorique* $E(P)$ d'un peptide P est l'ensemble des masses de ses peptides partiels. Un *spectre* (expérimental) $S = \{s_1, \ldots, s_m\}$ est un ensemble de

masses d'ions (fragments). Un *score* entre un spectre S et un peptide P est égal au nombre de masses que le spectre expérimental et le spectre théorique ont en commun. On est ainsi conduit à envisager le problème suivant :

Problème du séquençage peptidique Étant donnés le spectre S et une masse parente m, trouver un peptide de masse m ayant un score maximal pour le spectre S.

Chapitre 2

Cartographie de restriction

2.1 Introduction

Hamilton Smith a découvert en 1970 que l'*enzyme de restriction Hin*dII coupait les molécules d'ADN à chaque occurrence d'une séquence GTGCAC ou GTTAAC (Smith et Wilcox, 1970 [319]). Peu de temps après, Danna *et al.*, 1973 [80] construisaient la première carte de restriction pour l'ADN du virus simien 40. Depuis, les *cartes de restriction* (appelées aussi parfois *cartes physiques*) représentant les molécules d'ADN avec les points de clivage (des *sites*) obtenus grâce à des enzymes de restriction, sont devenues des structures de données fondamentales en biologie moléculaire.

Pour construire une carte de restriction, les biologistes utilisent différentes techniques *biochimiques* afin de trouver une information indirecte concernant la carte, ainsi que des méthodes *combinatoires* pour la reconstruire à partir de ces données. Il existe plusieurs approches expérimentales pour la cartographie de restriction, chacune présentant des avantages et des inconvénients. Elles mènent à différents problèmes combinatoires qu'il est souvent possible de formuler sous la forme de problèmes qui consistent à trouver les positions de points dont on ne connaît que quelques unes des distances qui les séparent.

La plupart des problèmes de cartographie de restriction correspondent au problème suivant. Si X est un ensemble de points alignés, on note ΔX le multi-ensemble formé de *toutes* les distances séparant les points de X : $\Delta X = \{|x_1 - x_2| : x_1, x_2 \in X\}$. En cartographie de restriction, on donne un sous-ensemble $E \subset \Delta X$ correspondant aux données expérimentales sur les longueurs du fragment et le problème est de reconstruire X à partir de E.

Pour le *problème de la digestion partielle* (PDP), l'expérimentation fournit des données sur *toutes* les distances entre les sites de restriction ($E = \Delta X$). Avec cette méthode, l'ADN est digéré de sorte que les fragments soient formés par toute paire de coupures. On ne connaît pas encore d'algorithme polynomial résolvant le PDP. La difficulté est qu'il est peut-être impossible de reconstruire X de façon unique à partir de ΔX : deux ensembles X et Y

sont dits *homométriques* s'ils vérifient $\Delta X = \Delta Y$. Par exemple, X, $-X$ (le réfléchi de X) et $X + a$ (le translaté de X), pour tout nombre a, sont homométriques. Il y a peu d'exemples d'ensembles homométriques non triviaux ; par exemple, les ensembles $\{0, 1, 3, 8, 9, 11, 13, 15\}$ et $\{0, 1, 3, 4, 5, 7, 12, 13, 15\}$ sont homométriques et on ne peut passer de l'un à l'autre par des réflexions et des translations (ce sont des *ensembles fortement homométriques*). Rosenblatt et Seymour, 1982 [289] ont étudié les ensembles fortement homométriques et ont donné un élégant algorithme pseudo-polynomial pour le PDP, fondé sur la factorisation polynomiale. Par la suite, Skiena *et al.*, 1990 [314] ont proposé un algorithme *marche arrière* simple, qui fonctionne très bien en pratique mais qui, dans certains cas, peut nécessiter un temps exponentiel.

L'algorithme marche arrière résout facilement le problème PDP pour toutes les données obtenues dans la pratique. Cependant, le PDP n'a jamais été la méthode de cartographie préférée dans les laboratoires de biologie car il est difficile de digérer l'ADN de façon à former des coupures entre *chaque* paire de sites.

La *double digestion* est une technique expérimentale de cartographie beaucoup plus simple que la digestion partielle. Dans cette approche, un biologiste cartographie les positions des sites de deux enzymes de restriction par la digestion complète de l'ADN, de sorte que seuls les fragments situés entre des sites *consécutifs* soient formés. Pour construire une telle carte, on peut mesurer les longueurs des fragments (et non pas l'ordre) d'une digestion complète de l'ADN par chacune des deux enzymes séparément, puis par les deux enzymes appliquées simultanément. Le problème consistant à déterminer les positions des coupures à partir des longueurs des fragments est connu sous le nom de *problème de la double digestion* (DDP pour *Double Digest Problem*).

Pour un ensemble quelconque X de n éléments, on note δX l'ensemble des $n-1$ distances entre éléments *consécutifs* de X. Dans le problème de la double digestion, un multi-ensemble $X \subset [0, t]$ est divisé en deux sous-ensembles $X = A \cup B$, avec $0 \in A \cap B$ et $t \in A \cap B$ et l'expérimentation fournit trois ensembles de longueurs : $\delta A, \delta B$ et δX (A et B correspondent aux digestions simples, tandis que X correspond à la double digestion). Le problème de la double digestion est de reconstruire A et B à partir de ces données.

Les premières tentatives de résolution du problème de la double digestion (Stefik, 1978 [329]) étaient loin d'être un succès. En effet, le nombre de cartes potentielles et la complexité des calculs du DDP croissent très rapidement avec le nombre de sites. Les erreurs expérimentales compliquent encore le problème. Les algorithmes DDP rencontrent des difficultés d'ordre calculatoire, même pour de petites cartes avec moins de dix sites pour chaque enzyme de restriction.

Goldstein et Waterman, 1987 [130] ont prouvé que le DDP est NP-complet et ont montré que le nombre de solutions du DDP augmente de façon exponentielle avec le nombre de sites. Malgré cela, Schmitt et Waterman, 1991 [309] ont noté que, même si le nombre de solutions augmente très rapidement avec le nombre de sites, la plupart des solutions sont très voisines (on peut passer de l'une à l'autre par des transformations simples). Comme les algorithmes de car-

tographie produisent bon nombre de « cartes très similaires », il semble raisonnable de diviser l'ensemble complet des cartes physiques en classes d'équivalence et de produire seulement une carte de base pour chaque classe d'équivalence. Par la suite, on obtient toutes les solutions en appliquant des transformations simples aux cartes de base. Si le nombre de classes d'équivalence était inférieur au nombre de cartes physiques de manière significative, alors cette approche permettrait de réduire le temps de calcul pour l'algorithme DDP.

Schmitt et Waterman, 1991 [309] ont fait le premier pas dans cette direction et ont introduit une relation d'équivalence sur l'ensemble des cartes physiques. Toutes les cartes de la même classe d'équivalence sont transformées les unes des autres par des *transformations de cassettes.* Le problème de la construction de toutes les classes d'équivalence pour le DDP restait néanmoins ouvert et on ne connaissait pas d'algorithme pour une transformation de cartes équivalentes. Pevzner, 1995 [267] a établi un théorème de caractérisation pour les transformations équivalentes de cartes physiques et a décrit la manière de produire toutes les solutions d'un problème DDP. Ce résultat est fondé sur les relations entre les solutions du DDP et les cycles eulériens dans des graphes aux arêtes coloriées.

Comme nous l'avons vu, les algorithmes combinatoires pour le PDP sont très rapides dans la pratique, mais les données expérimentales du PDP sont difficiles à obtenir. Inversement, les expériences pour le DDP sont très simples, mais les algorithmes combinatoires sont trop lents. C'est la raison pour laquelle la cartographie de restriction n'est pas une technique expérimentale très populaire de nos jours.

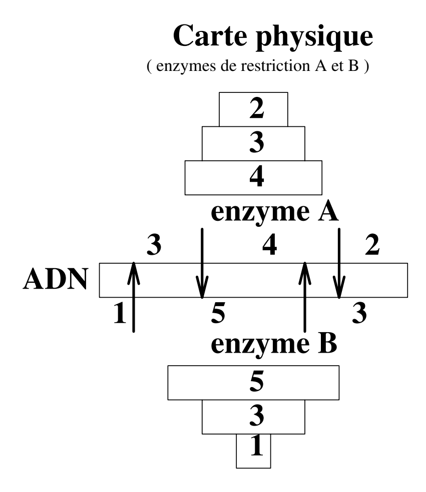


Figure 2.1 – Carte physique de deux enzymes de restriction. L'électrophorèse sur gel fournit de l'information sur les tailles (mais pas sur l'ordre) des fragments de restriction.

2.2 Problème de la double digestion

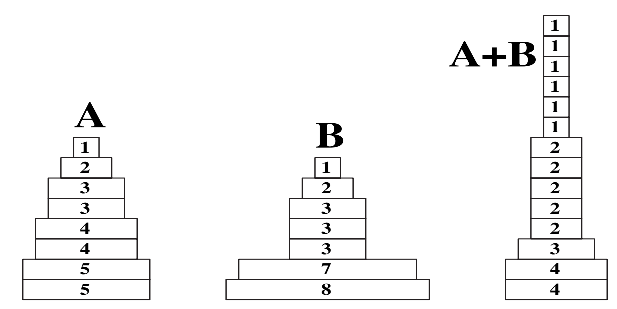
La figure 2.1 montre de l'ADN coupé par des enzymes de restriction A et B. Lorsque Danna *et al.*, 1973 [80] ont construit la première carte physique, il n'existait pas de technique expérimentale permettant de trouver directement les positions des coupures. Cependant, ils ont pu mesurer les tailles (mais pas l'ordre !) des fragments de restriction à l'aide d'une technique expérimentale connue sous le nom d'*électrophorèse sur gel*. Grâce à des expériences d'électrophorèse sur gel avec deux enzymes de restriction A et B (figure 2.1), un biologiste obtient de l'information sur les tailles des fragments de restriction 2, 3, 4 pour A et 1, 3, 5 pour B, mais il existe de nombreux ordonnancements (cartes) correspondant à ces tailles (la figure 2.2 en montre deux). Pour trouver quelle carte de la figure 2.2 convient, les biologistes utilisent la *double digestion* $A + B$ (clivage de l'ADN par *les deux* enzymes A et B). Les deux cartes présentées dans la figure 2.2 produisent les mêmes digestions simples A et B mais des doubles digestions $A + B$ différentes $(1, 1, 2, 2, 3$ et $1, 1, 1, 2, 4)$. La double digestion qui convient aux données expérimentales correspond à la carte correcte. Le problème de la double digestion consiste à trouver une carte physique, étant données les trois piles de fragments : A, B et $A + B$ (figure 2.3).

Figure 2.2 – Les données sur A et B ne permettent pas à un biologiste de trouver une carte exacte. Les données sur $A + B$ aident à trouver une carte correcte.

2.3 Solutions multiples au problème de la double digestion

La figure 2.3 présente deux solutions au problème de la double digestion. Bien qu'elles semblent très différentes, on peut passer de l'une à l'autre par une opération simple appelée *changement de cassettes* (figure 2.4). La figure 2.5 donne un autre exemple de solutions multiples. Bien qu'on ne puisse pas passer de l'une à l'autre par changement de cassettes, elles peuvent être transformées l'une en l'autre grâce à une opération différente appelée *réflexion de cassette*

(figure 2.6). Un résultat surprenant est que ces deux opérations simples (en un certain sens) sont suffisantes pour permettre le passage entre solutions semblables au problème de la double digestion.



Problème de la double digestion :
étant données A, B et A+B, trouver une carte physique

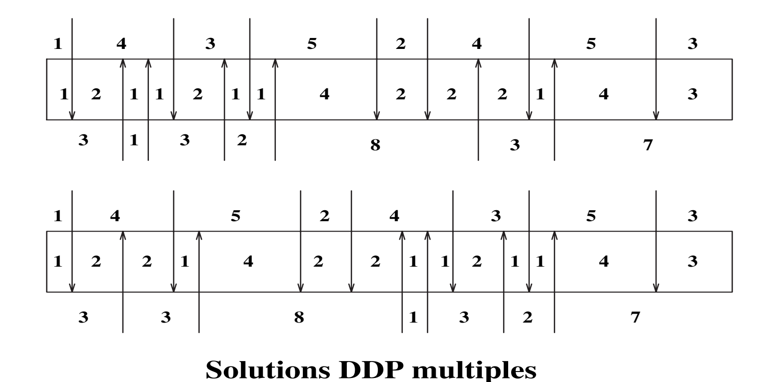


Figure 2.3 – Le problème de la double digestion peut avoir des solutions multiples.

Une carte physique est représentée par la séquence ordonnée des fragments des digestions simples $A_1, \ldots, A_n$ et $B_1, \ldots, B_m$ et de la double digestion $C_1, \ldots, C_l$ (figure 2.4). Pour un intervalle entier $I = [i, j]$ avec $1 \leq i \leq j \leq l$, on définit $I_C = \{C_k : \ i \leq k \leq j\}$ comme étant l'ensemble des fragments situés entre C_i et C_j. La *cassette* définie par I_C est la paire d'ensembles de fragments (I_A, I_B), où I_A et I_B sont les ensembles de tous les fragments respectivement de A et de B qui contiennent un fragment de I_C (figure 2.4). Soient m_A et m_B les positions de départ des fragments situés le plus à gauche de I_A et de I_B respectivement. Le *chevauchement gauche* de (I_A, I_B) est la distance $m_A - m_B$. Le *chevauchement droit* de (I_A, I_B) est défini de façon similaire.

Supposons que deux cassettes dans la solution au DDP aient les mêmes chevauchements gauches et les mêmes chevauchements droits. Si ces cassettes sont disjointes, alors elles peuvent être *échangées* comme dans la figure 2.4 et l'on obtient une nouvelle solution au DDP. De la même façon, si les chevauchements

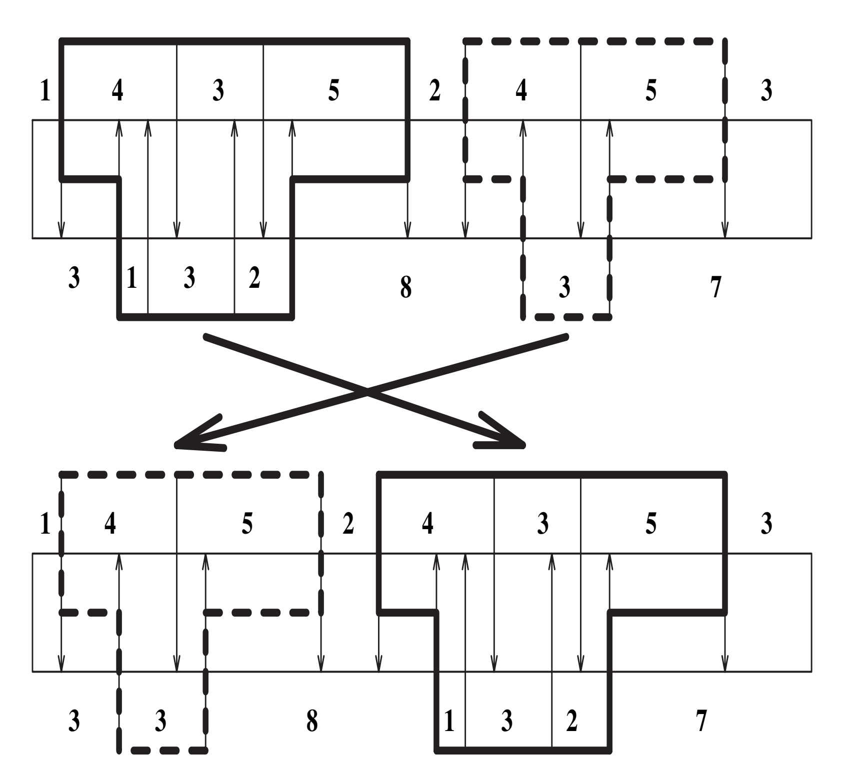


Figure 2.4 – Échange de cassettes. La carte du haut est définie par les séquences ordonnées des tailles des fragments pour l'enzyme de restriction A ($\{1, 4, 3, 5, 2, 4, 5, 3\}$), l'enzyme de restriction B ($\{3, 1, 3, 2, 8, 3, 7\}$) et les enzymes de restriction $A + B = C$ ($\{1, 2, 1, 1, 2, 1, 1, 4, 2, 2, 2, 1, 4, 3\}$). L'intervalle $I = [3, 7]$ définit l'ensemble des fragments de la double digestion $I_C = \{C_3, C_4, C_5, C_6, C_7\}$ de longueurs 1, 1, 2, 1, 1. L'ensemble I_C définit une cassette (I_A, I_B), où $I_A = \{A_2, A_3, A_4\} = \{4, 3, 5\}$ et $I_B = \{B_2, B_3, B_4\} = \{1, 3, 2\}$. Le chevauchement gauche de (I_A, I_B) vaut $m_A - m_B = 1 - 3 = -2$. Le chevauchement droit de (I_A, I_B) est égal à $13 - 9 = 4$.

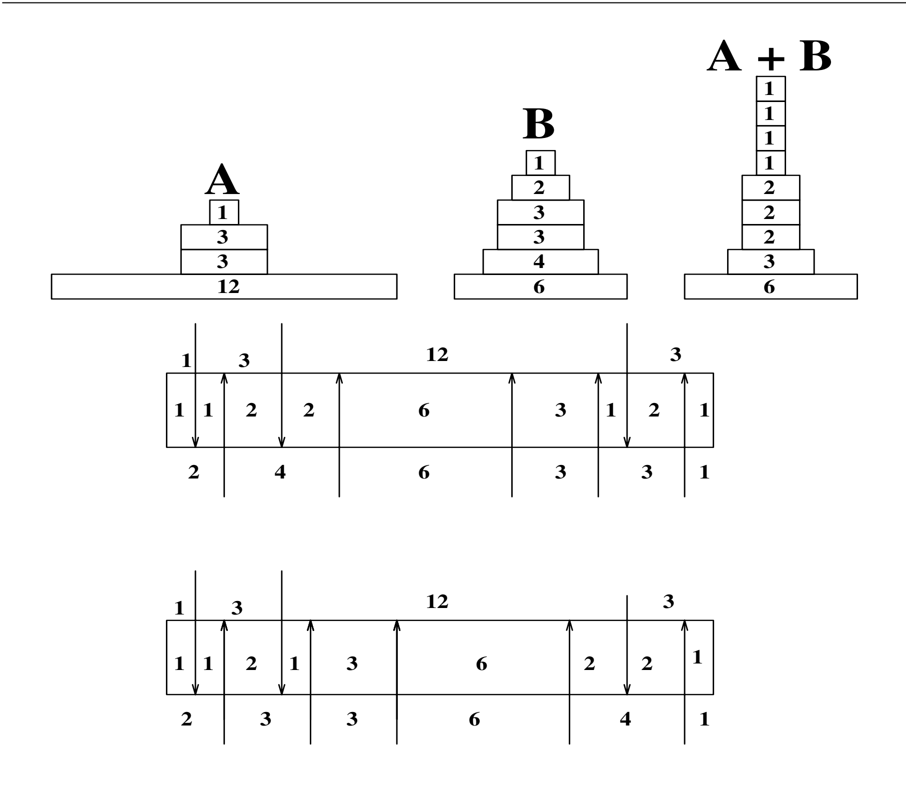


Figure 2.5 – Des solutions DDP multiples ne pouvant être transformées l'une en l'autre par un échange de cassette.

gauches et droits d'une cassette (I_A, I_B) ont la même taille mais des signes différents, alors la cassette peut être *réfléchie*, comme le montre la figure 2.6, et l'on obtient une nouvelle solution au DDP.

Schmitt et Waterman, 1991 [309] ont posé la question suivante : comment transformer une carte en une autre par des transformations de cassettes ? La partie suivante introduit une méthode issue de la théorie des graphes pour analyser la combinatoire des transformations de cassettes et répondre à la question de Schmitt-Waterman.

2.4 Cycles alternés dans les graphes coloriés

Considérons un *graphe non orienté* $G(V, E)$ dont chaque *arête* est coloriée en une *couleur* parmi l possibles. Une suite de sommets $P = x_1 x_2 \ldots x_m$ est appelée un *chemin* dans G si, pour tout $1 \leq i \leq m - 1$, (x_i, x_{i+1}) est une

Figure 2.6 – Réflexion de cassette. Les chevauchements gauches et droits ont la même taille mais des signes différents.

arête de E. Un chemin P est appelé un *cycle* si x_1 et x_m sont confondus. Les chemins et les cycles peuvent se couper en un ou plusieurs sommets. On introduit $P^- = x_m x_{m-1} \dots x_1$.

Un chemin (ou un cycle) dans G est dit *alterné* s'il ne possède pas deux arêtes consécutives de même couleur (si P est un cycle, on considère que (x_{m-1}, x_m) et (x_1, x_2) sont des arêtes consécutives). Un chemin (ou un cycle) P dans G est dit *eulérien* si chaque arête $e \in E$ est traversée par P exactement une fois. Soient $d_c(v)$ le nombre d'arêtes de couleur c de E incidentes au sommet v et $d(v) = \sum_{c=1}^{l} d_c(v)$ le *degré* de v dans le graphe G. Un sommet v dans le graphe G est dit *équilibré* si $\max_c d_c(v) \leq d(v)/2$. Un *graphe équilibré* est un graphe dont chaque sommet est équilibré.

Théorème 2.1 *(Kotzig, 1968 [206]) Soit G un graphe connexe colorié avec des sommets de degrés pairs. Il existe un cycle eulérien alterné dans G si et seulement si G est équilibré.*

Preuve Pour construire un cycle eulérien alterné dans G, on partitionne les $d(v)$ arêtes incidentes au sommet v en $d(v)/2$ paires, de telle sorte que deux arêtes de la même paire aient des couleurs différentes (ceci peut être réalisé pour tout sommet équilibré). En commençant par une arête quelconque dans G, on forme une traînée C_1 en utilisant à chaque étape une arête appariée à la dernière arête de la traînée. Le processus s'arrête lorsqu'une arête appariée à la dernière de la traînée a déjà été utilisée dans celle-ci. Comme chaque sommet

dans G est de degré pair, chaque traînée de ce type qui démarre en un sommet v se termine en v. Avec un peu de chance, la traînée sera eulériennne ; si ce n'est pas le cas, elle doit contenir un sommet w ayant encore un certain nombre d'arêtes non traversées. Comme le graphe formé par les arêtes non traversées est équilibré, on peut commencer en w et former une autre traînée C_2 composée d'arêtes non traversées en utilisant la même règle. On peut désormais combiner C_1 et C_2 comme suit : on insère la traînée C_2 dans C_1 là où w est atteint. Pour cela, on doit procéder avec précaution, afin de préserver l'alternance des couleurs au niveau du sommet w. On peut voir que, si l'insertion directe de la traînée C_2 détruit l'alternance de couleurs, alors il suffit de l'insérer dans l'ordre inverse pour préserver cette propriété. En répétant ceci, on obtient un cycle eulérien alterné. ∎

Nous allons utiliser le corollaire suivant du théorème de Kotzig :

Lemme 2.1 *Soit G un graphe connexe bicolore. Alors il existe un cycle eulérien alterné dans G si et seulement si $d_1(v)$ est égal à $d_2(v)$ pour tout sommet v de G.*

2.5 Transformations de cycles eulériens alternés

Dans cette section, on introduit des *transformations d'ordre* de chemins alternés et l'on démontre que, pour toute paire de cycles eulériens alternés dans un graphe bicolore, on peut passer de l'un à l'autre par des transformations d'ordre. Ce résultat conduit à la caractérisation de Schmitt-Waterman des transformations de cassettes.

Soit $F = \ldots x \ldots y \ldots x \ldots y \ldots$ un chemin alterné dans un graphe bicolore G. Les sommets x et y scindent F en cinq sous-chemins $F = F_1F_2F_3F_4F_5$ (voir figure 2.7). La transformation $F = F_1F_2F_3F_4F_5 \longrightarrow F^* = F_1F_4F_3F_2F_5$ est appelée un *changement d'ordre* si F^* est un chemin alterné.

Soit $F = \ldots x \ldots x \ldots$ un chemin alterné dans un graphe bicolore G. Un sommet x partage F en trois sous-chemins $F = F_1F_2F_3$ (voir figure 2.8). La transformation $F = F_1F_2F_3 \longrightarrow F^* = F_1\bar{F}_2F_3$ est appelée une *réflexion d'ordre* si F^* est un chemin alterné. Bien évidemment, la réflexion d'ordre $F \longrightarrow F^*$ dans un graphe bicolore existe si et seulement si F_2 est un cycle impair.

Théorème 2.2 *Pour tout couple de cycles eulériens alternés d'un graphe bicolore G, on peut transformer l'un en l'autre par une série de transformations d'ordre (changements ou réflexions).*

Preuve Soient X et Y deux cycles eulériens alternés dans G. Considérons l'ensemble des cycles eulériens alternés $\mathcal{C}$ obtenus à partir de X par toutes les séries possibles de transformations d'ordre. Soit $X^* = x_1 \ldots x_m$ un cycle de $\mathcal{C}$ ayant le plus long préfixe commun avec $Y = y_1 \ldots y_m$. Dans ce cas on a

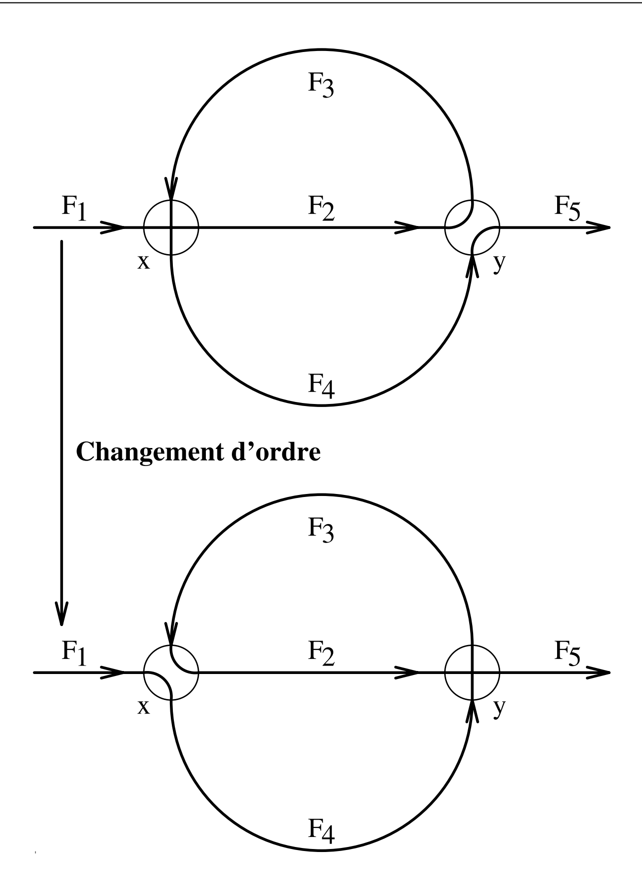


Figure 2.7 – Changement d'ordre.

$x_1 \ldots x_l = y_1 \ldots y_l$ pour $l \leq m$. Si l est égal à m, le théorème est vrai ; sinon, on pose $v = x_l = y_l$ (autrement dit, $e_1 = (v, x_{l+1})$ et $e_2 = (v, y_{l+1})$ sont les premières arêtes différentes dans X^* et Y, respectivement (voir figure 2.9)).

Comme X^* et Y sont des chemins alternés, les arêtes e_1 et e_2 ont la même couleur. Puisque X^* est un chemin eulérien, X^* contient l'arête e_2. Il est clair que e_2 suit e_1 dans X^*. Deux cas (voir figure 2.9) se présentent selon la direction de l'arête e_2 dans le chemin X^* (dirigée vers v ou issue de v) :

Premier cas. L'arête $e_2 = (y_{l+1}, v)$ dans le chemin X^* est dirigée vers v. Dans ce cas, on a $X^* = x_1 \ldots v\ x_{l+1} \ldots y_{l+1}\ v \ldots x_m$. Comme les couleurs des arêtes e_1 et e_2 coïncident, la transformation $X^* = F_1F_2F_3 \longrightarrow F_1F_2^-F_3 = X^{**}$ est une réflexion d'ordre (figure 2.10). Par conséquent, on a $X^{**} \in \mathcal{C}$ et au moins $(l+1)$ sommets initiaux de X^{**} et de Y coïncident, ce qui contredit le choix de X^*.

Second cas. L'arête $e_2 = (v, y_{l+1})$ du chemin X^* part de v. Dans ce cas, le sommet v partage le chemin X^* en trois parties : un préfixe X_1 qui se termine en v, un cycle X_2 et un suffixe X_3 qui débute en v. Il est facile de voir que X_2

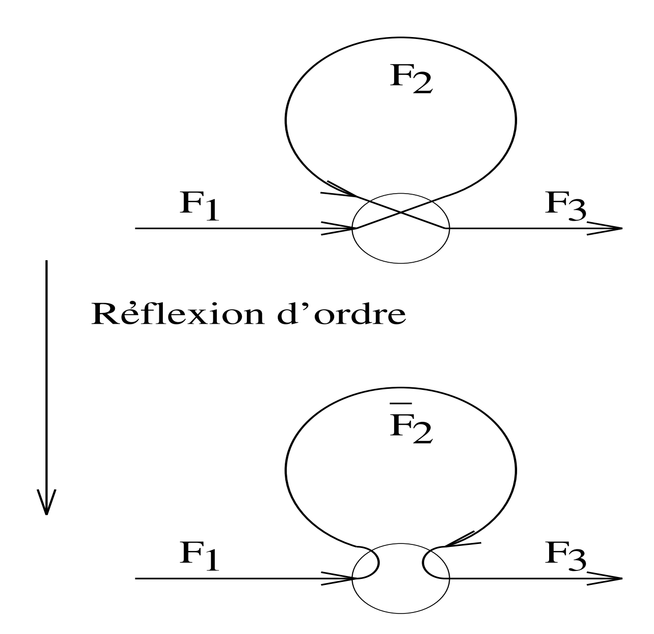


Figure 2.8 – Réflexion d'ordre.

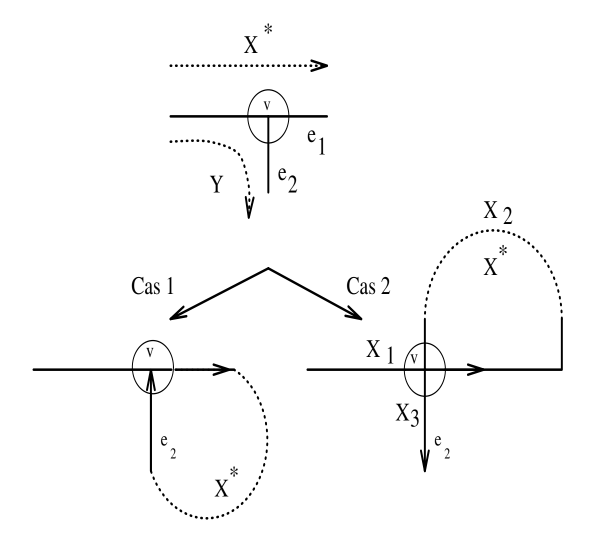


Figure 2.9 – Deux cas pour le théorème 2.2.

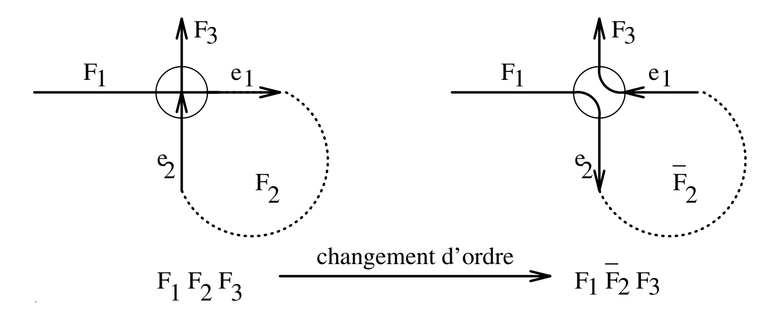


Figure 2.10 – Premier cas : changement d'ordre.

et X_3 ont un sommet commun $x_j = x_k$ $(l < j < k < m)$ (sinon, Y ne serait pas un cycle eulérien). Par conséquent, le cycle X^* peut désormais être réécrit sous la forme $X^* = F_1F_2F_3F_4F_5$ (voir figure 2.11).

Considérons les arêtes (x_k, x_{k+1}) et (x_{j-1}, x_j), représentées en gras dans la figure 2.11. Si elles sont de couleurs différentes, alors $X^{**} = F_1F_4F_3F_2F_5$ est le cycle alterné obtenu à partir de X^* après le changement d'ordre exhibé dans la figure 2.11 (en haut). Dans ce cas, au moins $(l+1)$ sommets initiaux de X^{**} et de Y coïncident, ce qui contredit le choix de X^*.

Si les arêtes (x_k, x_{k+1}) et (x_{j-1}, x_j) sont de la même couleur (figure 2.11, en bas), alors $X^{**} = F_1F_4F_2^-F_3^-F_5$ est obtenu à partir de X^* par deux réflexions d'ordre g et h :

$$F_1F_2F_3F_4F_5 \xrightarrow{g} F_1F_2(F_3F_4)^-F_5 = F_1F_2F_4^-F_3^-F_5 \xrightarrow{h}$$

$$F_1(F_2F_4^-)^-F_3^-F_5 = F_1F_4^{--}F_2^-F_3^-F_5 = F_1F_4F_2^-F_3^-F_5$$

Dans ce cas, au moins $(l+1)$ sommets initiaux de X^{**} et de Y coïncident, ce qui contredit le choix de X^*. ■

2.6 Cartes physiques et cycles eulériens alternés

Cette section introduit les *graphes de bifurcation* des cartes physiques et démontre que toute carte physique correspond à un chemin eulérien alterné dans le graphe de bifurcation.

Considérons une carte physique définie par des fragments (ordonnés) de digestions simples A et B et la double digestion $C = A + B$: $\{A_1, \ldots, A_n\}$, $\{B_1, \ldots, B_m\}$ et $\{C_1, \ldots, C_l\}$. Dans un souci de simplicité, on supposera dorénavant que A et B ne coupent pas l'ADN aux mêmes endroits, c'est-à-dire $l = n + m - 1$. Une *bifurcation* d'un fragment A_i est l'ensemble des fragments de la double digestion C_j contenus dans A_i :

$$F(A_i) = \{C_j : \quad C_j \subset A_i\}$$

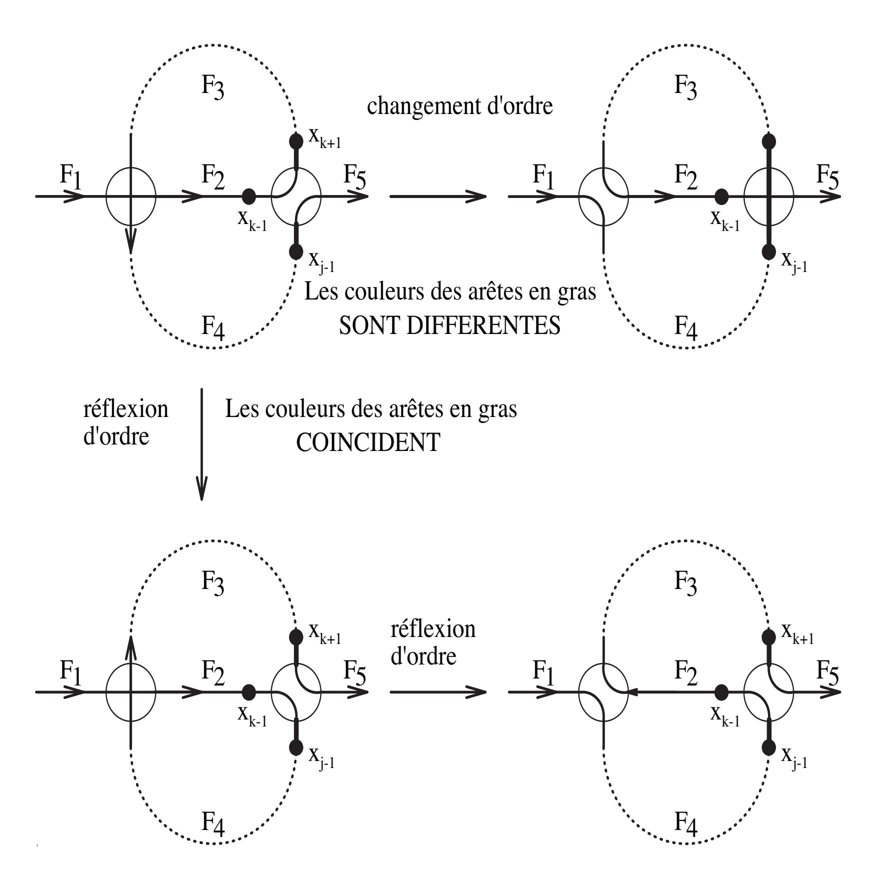


Figure 2.11 – Second cas : suivant les couleurs des arêtes en gras, il existe soit un changement d'ordre, soit deux réflexions d'ordre transformant X^* en un cycle dont le préfixe commun avec Y est plus long que celui de X^*.

(une bifurcation de B_i se définit de façon analogue). Dans l'exemple présenté en figure 2.12, $F(A_3)$ est constitué de deux fragments C_5 et C_6 de tailles respectives 4 et 1. Évidemment, toute paire de bifurcations $F(A_i)$ et $F(B_j)$ a au plus un fragment commun. Une bifurcation contenant au moins deux fragments est appelée une *multi-bifurcation*.

Les fragments les plus à gauche et les plus à droite des multi-bifurcations sont appelés des *fragments frontières*. Évidemment, C_1 et C_l sont des fragments frontières.

Lemme 2.2 *À l'exception de C_1 et de C_l, tout fragment frontière appartient à exactement deux multi-bifurcations $F(A_i)$ et $F(B_j)$. Les fragments frontières C_1 et C_l appartiennent exactement à une multi-bifurcation.*

Le lemme 2.2 motive la construction du *graphe de bifurcation* dont l'ensemble des sommets est constitué des longueurs des fragments frontières (deux

fragments frontières de même longueur correspondent au même sommet). L'ensemble des arêtes du graphe de bifurcation correspond à toutes les multi-bifurcations (chacune d'elles est représentée par une arête qui relie les sommets correspondant à la longueur de ses fragments frontières). Les arêtes correspondant aux multi-bifurcations de A reçoivent la couleur A, tandis que les arêtes correspondant aux multi-bifurcations de B ont la couleur B (figure 2.12).

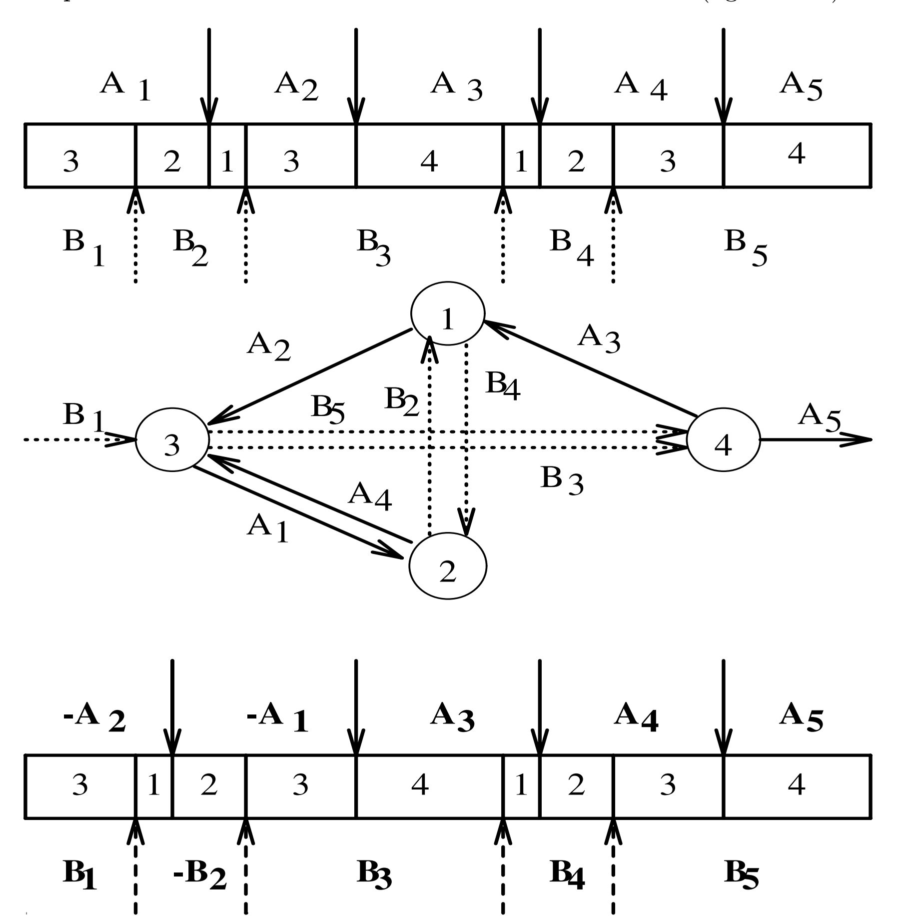


Figure 2.12 – Le graphe de bifurcation d'une carte physique où l'on a ajouté les arêtes B_1 et A_5. Les arêtes dessinées en traits pleins (resp. en pointillés) correspondent aux multi-bifurcations de A (resp. B). Les flèches sur les arêtes de ce graphe (non orienté) suivent le chemin $B_1A_1B_2A_2B_3A_3B_4A_4B_5A_5$, qui correspond à la carte du haut. La carte du bas, $B_1(-A_2)(-B_2)(-A_1)B_3A_3B_4A_4B_5A_5$, est obtenue en changeant l'orientation des arêtes dans le triangle A_1, B_2, A_2 (réflexion de cassette).

Tous les sommets de G sont équilibrés, sauf peut-être les sommets $|C_1|$ et $|C_l|$ qui sont semi-équilibrés (autrement dit, ils vérifient $|d_A(|C_1|) - d_B(|C_1|)| = |d_A(|C_l|) - d_B(|C_l|)| = 1$). Le graphe G peut être transformé en un graphe

équilibré en ajoutant une ou deux arête(s). Par conséquent, G contient un chemin eulérien alterné.

Toute carte physique (A, B) définit un chemin eulérien alterné dans son graphe de bifurcation. Les transformations de cassettes d'une carte physique ne changent pas l'ensemble des bifurcations de cette carte. On est alors en droit de se poser la question suivante : étant données deux cartes ayant le même ensemble de bifurcations, peut-on passer de l'une à l'autre par des transformations de cassettes ? La figure 2.12 présente deux cartes ayant le même ensemble de bifurcations ; elles correspondent à deux cycles eulériens alternés dans le graphe de bifurcation. Il est facile de voir que les transformations de cassettes des cartes physiques correspondent aux transformations d'ordre dans le graphe de bifurcation. Par conséquent, tout chemin eulérien alterné dans le graphe de bifurcation de (A, B) correspond à une carte obtenue à partir de (A, B) par des transformations de cassettes (théorème 2.2).

2.7 Problème de la digestion partielle

Le problème de la digestion partielle consiste à reconstruire les positions de n sites de restriction à partir de l'ensemble des C_n^2 distances qui les séparent. Si ΔX désigne l'ensemble des distances entre toutes les paires de points de X, alors le problème PDP consiste à reconstruire X à partir de ΔX. Rosenblatt et Seymour, 1982 [289] ont donné un algorithme pseudo-polynomial pour ce problème, qui utilise la factorisation de polynômes. Skiena *et al.*, 1990 [314] ont décrit le simple algorithme marche arrière ci-dessous, modifié ultérieurement par Skiena et Sundaram, 1994 [315] dans le cas de données entachées d'erreurs.

Il faut d'abord trouver la plus longue distance dans ΔX, qui décide des deux points situés aux extrémités de X, puis supprimer cette distance de ΔX. Ensuite, on positionne au fur et à mesure la distance restante la plus longue de ΔX. Comme, à chaque étape, la distance la plus longue dans ΔX doit être réalisée par l'un des points des extrémités, il n'y a que deux positions possibles (gauche ou droite) pour placer le point. À chaque étape et pour chacune des deux positions, on regarde si toutes les distances entre cette position et les points déjà sélectionnés sont dans ΔX. Si c'est le cas, on supprime toutes ces distances avant de passer à l'étape suivante. Si ce n'est le cas d'aucune des deux positions, on fait marche arrière. On a trouvé une solution lorsque ΔX est vide.

Par exemple, supposons que l'on ait $\Delta X = \{2, 2, 3, 3, 4, 5, 6, 7, 8, 10\}$. Comme ΔX contient toutes les distances entre deux points distincts, on a : $|\Delta X| = C_n^2$, où n est le nombre de points dans la solution. On en déduit donc que n vaut 5 et on pose $X = \{x_1, x_2, \ldots, x_5\}$, avec $x_1 \leq x_2 \leq \ldots \leq x_5$. Soient $L = \Delta X$ et $x_1 = 0$. Comme 10 est la plus grande distance dans L, il est clair que x_5 vaut 10. En enlevant la distance $x_5 - x_1 = 10$ de L, on obtient :

$$X = \{0, 10\} \quad \text{et} \quad L = \{2, 2, 3, 3, 4, 5, 6, 7, 8\}.$$

La plus grande distance restante est 8. On a désormais deux choix possibles : soit $x_4 = 8$, soit $x_2 = 2$. Comme ces deux cas sont miroirs l'un de l'autre, on peut, sans perte de généralité, supposer : $x_2 = 2$. Après avoir enlevé les distances $x_5 - x_2 = 8$ et $x_2 - x_1 = 2$ de L, on obtient :

$$X = \{0, 2, 10\} \quad \text{et} \quad L = \{2, 3, 3, 4, 5, 6, 7\}.$$

Comme 7 est la plus grande distance restante, on a soit $x_4 = 7$, soit $x_3 = 3$. Si l'on choisit $x_3 = 3$, la distance $x_3 - x_2 = 1$ devrait être dans L, ce qui n'est pas le cas ; on ne peut donc que poser $x_4 = 7$. Après avoir enlevé les distances $x_5 - x_4 = 3$, $x_4 - x_2 = 5$ et $x_4 - x_1 = 7$ de L, on obtient :

$$X = \{0, 2, 7, 10\} \quad \text{et} \quad L = \{2, 3, 4, 6\}.$$

Désormais, 6 est la plus grande distance restante. On a de nouveau deux possibilités : soit $x_3 = 4$, soit $x_3 = 6$. Si l'on choisit $x_3 = 6$, la distance $x_4 - x_3 = 1$ devrait être dans L, ce qui n'est pas le cas. On doit donc prendre $x_3 = 4$ et ceci nous donne une solution $\{0, 2, 4, 7, 10\}$ au problème de la digestion partielle.

Le pseudo-code de l'algorithme suivi dans l'exemple est donné ci-dessous. La fonction **Supprime_Max**(L) retourne la valeur maximale de L et la retire de la liste L ; on utilise deux variables globales X et *largeur*. $\Delta(X, Y)$ est l'ensemble constitué de toutes les distances entre un point de X et un point de Y.

```
int largeur
Digestion_Partielle(Liste L)
    largeur = Supprime_Max(L)
    X = {0, largeur}
    Placer(L)

Placer(Liste L)
    si L = ∅ alors
        donner la solution X
        sortie
    y = Supprime_Max(L)
    si Δ({y}, X) ⊂ L alors
        X = X ∪ {y}
        Placer(L \ Δ({y}, X))       /* place un point à droite */
        X = X \ {y}                 /* marche arrière */
    si Δ({largeur − y}, X) ⊂ L alors
        X = X ∪ {largeur − y}
        Placer(L \ Δ({largeur − y}, X)) /* place un point à gauche */
        X = X \ {largeur − y}           /* marche arrière */
```

Cet algorithme se déroule en un temps $O(n^2 \log n)$ si L provient de points réels dans des positions générales car, dans ce cas, à chaque étape, l'un des

deux choix possibles sera valable avec une probabilité 1. Cependant, la complexité temporelle de l'algorithme est exponentielle dans le pire des cas (Zhang, 1994 [377]).

2.8 Ensembles homométriques

Il n'est pas toujours possible de reconstruire un ensemble X de façon unique à partir de ΔX. Des ensembles A et B sont dits *homométriques* s'ils vérifient $\Delta A = \Delta B$. Soient I et J deux multi-ensembles. Il est facile de vérifier que les multi-ensembles $I + J = \{i + j : i \in I, j \in J\}$ et $I - J = \{i - j : i \in I, j \in J\}$ sont homométriques. L'exemple présenté dans la figure 2.13 montre ce résultat pour $I = \{6, 7, 9\}$ et $J = \{-6, 2, 6\}$.

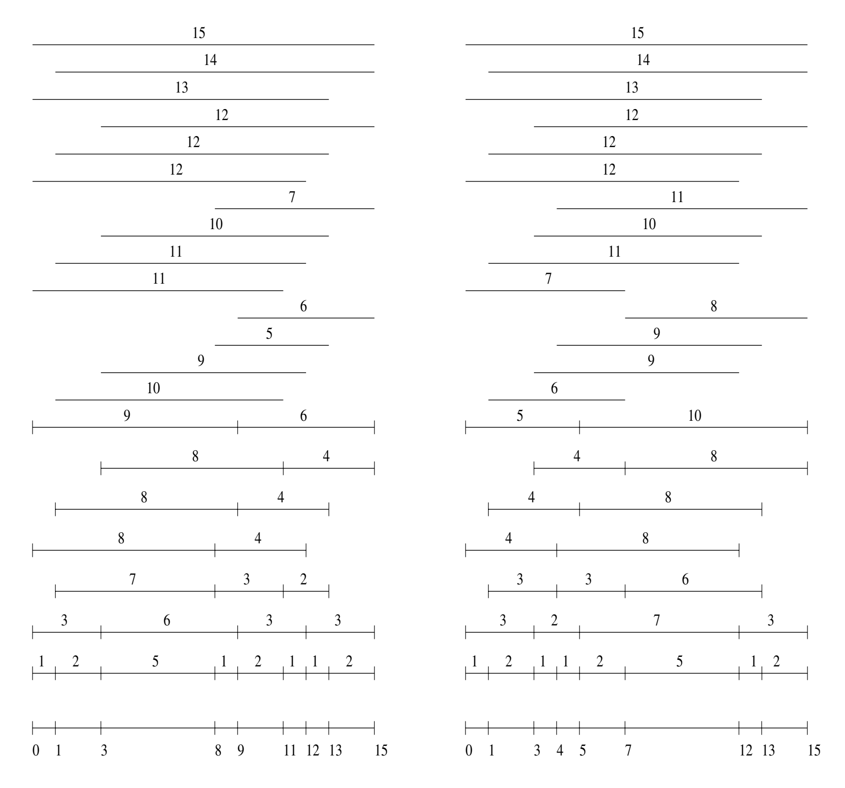


Figure 2.13 – Les ensembles $I + J = \{0, 1, 3, 8, 9, 11, 12, 13, 15\}$ et $I - J = \{0, 1, 3, 4, 5, 7, 12, 13, 15\}$ sont homométriques.

Il est alors naturel de se demander si toute paire d'ensembles homométriques provient d'une telle construction. La réponse est négative : les ensembles homométriques $\{0, 1, 2, 5, 7, 9, 12\}$ et $\{0, 1, 5, 7, 8, 10, 12\}$ fournissent un contre-exemple. Néanmoins, la conjecture précédente est vraie « si I et J peuvent admettre aussi des valeurs négatives ».

Étant donné un multi-ensemble d'entiers $A = \{a_i\}$, soit $A(x) = \sum_i x^{a_i}$ la *fonction génératrice* de A. Il est facile de voir que la fonction génératrice de ΔA est $\Delta A(x) = A(x)A(x^{-1})$. Soient $A(x)$ et $B(x)$ les fonctions génératrices des multi-ensembles A et B, avec $A(x) = U(x)V(x)$ et $B(x) = U(x)V(x^{-1})$. Alors on a : $A(x)A(x^{-1}) = B(x)B(x^{-1}) = U(x)V(x)U(x^{-1})V(x^{-1})$, ce qui implique que A et B sont homométriques.

Théorème 2.3 *(Rosenblatt et Seymour, 1982 [289]) Deux ensembles A et B sont homométriques si et seulement s'il existe des fonctions génératrices $U(x)$ et $V(x)$, ainsi qu'un entier β vérifiant $A(x) = U(x)V(x)$ et $B(x) = \pm x^{\beta}U(x)V(x^{-1})$.*

Preuve Soient A et B des ensembles homométriques. Soit $P(x)$ le plus grand diviseur commun à $A(x)$ et $B(x)$: écrivons $A(x) = P(x)Q_A(x)$ et $B(x) = P(x)Q_B(x)$, où $Q_A(x)$ et $Q_B(x)$ sont premiers entre eux. Soit $V(x)$ le plus grand diviseur commun à $Q_A(x)$ et $Q_B(x^{-1})$: écrivons $Q_A(x) = V(x)S_A(x)$ et $Q_B(x^{-1}) = V(x)S_B(x)$, où $S_A(x)$ et $S_B(x)$ sont premiers entre eux. Il est clair que $S_A(x)$ et $S_A(x^{-1})$ sont chacuns premiers avec $S_B(x)$ *et* $S_B(x^{-1})$.

Comme A et B sont homométriques, on obtient :

$$P(x)V(x)S_A(x)P(x^{-1})V(x^{-1})S_A(x^{-1}) = P(x)V(x^{-1})S_B(x^{-1})P(x^{-1})V(x)S_B(x),$$

ce qui implique : $S_A(x)S_A(x^{-1}) = S_B(x)S_B(x^{-1})$. Puisque $S_A(x)$ et $S_A(x^{-1})$ sont premiers avec $S_B(x)$ et $S_B(x^{-1})$, on a : $S_A(x) = \pm x^a$ et $S_B(x) = \pm x^b$. Par conséquent, on a : $A(x) = \pm x^a P(x)V(x)$ et $B(x) = \pm x^b P(x)V(x^{-1})$. La substitution $U(x) = \pm x^a P(x)$ achève la preuve du théorème. ■

Il peut arriver que certains coefficients dans la décomposition du théorème 2.3 soient négatifs ; ils correspondent aux multi-ensembles ayant des multiples négatifs. Par exemple, si l'on prend $A = \{0, 1, 2, 5, 7, 9, 12\}$ et $B = \{0, 1, 5, 7, 8, 10, 12\}$, on obtient $U(x) = (1 + x + x^2 + x^3 + x^4 + x^5 + x^7)$ et $V(x) = x^{-5}(1 - x^3 + x^5)$.

Un ensemble A est dit *reconstructible* si, lorsque B est homométrique à A, on a $B = A + \{v\}$ ou $B = -A + \{v\}$ pour un certain nombre v. Un ensemble A est dit *symétrique* s'il existe un nombre v qui vérifie $-A = A + v$. Un polynôme $A(x)$ est dit *symétrique* si l'ensemble correspondant est symétrique, c'est-à-dire $A(x^{-1}) = x^v A(x)$. Le théorème 2.3 implique le résultat suivant :

Théorème 2.4 *Un ensemble A est reconstructible si et seulement si $A(x)$ possède au moins un facteur premier (en comptant les multiples) non symétrique.*

Rosenblatt et Seymour, 1982 [289] ont proposé l'algorithme pseudo-polynomial suivant pour le problème de la digestion partielle avec n points. Étant donné un ensemble de C_n^2 distances $\Delta A = \{d_i\}$, on forme la fonction génératrice $\Delta A(x) = n + \sum_i (x^{d_i} + x^{-d_i})$. On factorise ce polynôme en produit de facteurs irréductibles dans l'anneau des polynômes à coefficients entiers à l'aide

d'un algorithme de factorisation de complexité polynomiale en $\max_i d_i$. La solution $A(x)$ au problème PDP doit être de la forme $\Delta A(x) = A(x)A(x^{-1})$. On essaie donc tous les 2^F produits possibles parmi les F facteurs irréductibles non réciproques* de $\Delta A(x)$. On trouve ainsi un ensemble formé de tous les ensembles $A(x)$ possibles. Finalement, on élimine les ensembles $A(x)$ ayant des coefficients négatifs, ainsi que les éventuels doublons.

2.9 Quelques autres problèmes et approches

2.9.1 Cartographie optique

Schwartz *et al.*, 1993 [311] ont développé la technique de la *cartographie optique* pour la construction des cartes de restriction. En cartographie optique, les copies simples des molécules d'ADN sont étirées et attachées à un support en verre sous un microscope. Quand les enzymes de restriction sont activées, elles fendent les molécules d'ADN en leurs sites de restriction. Les molécules restent attachées à la surface, mais l'élasticité de l'ADN tendu décroche les extrémités de la molécule aux lieux des sites clivés. Ces derniers peuvent être identifiés au microscope comme étant des petits trous dans la ligne fluorescente de la molécule. Une « photographie » de la molécule d'ADN présentant des trous aux positions des sites de clivage donne donc un instantané de la carte de restriction.

La cartographie optique contourne le problème de reconstruction de l'ordre des fragments de restriction mais fait apparaître de nouveaux challenges informatiques. Le problème est que tous les sites ne sont pas clivés dans chaque molécule (faux négatif) et que certains peuvent apparaître de façon incorrecte pour être coupés (faux positif). De plus, les imprécisions dans la mesure de la longueur des fragments, les difficultés dans l'analyse proximale des sites de restriction et l'orientation inconnue de chaque molécule (de gauche à droite ou vice versa) font de la reconstruction un processus difficile. En pratique, les données provenant de nombreuses molécules sont assemblées pour construire une carte de restriction de consensus.

Le problème de l'orientation inconnue a été formalisé par Muthukrishnan et Parida, 1997 [243] comme étant le problème de *retournement-coupure binaire* (BFC pour *Binary Flip-Cut*). Dans le problème BFC, on donne un ensemble de n chaînes binaires 0-1 (chaque chaîne représente un instantané d'une molécule d'ADN où les 1 correspondent aux sites de restriction). Le problème est d'associer à chaque chaîne un état de *retournement* ou de *non-retournement*, de telle sorte que le nombre de sites de *consensus* soit minimal. Un site est appelé un consensus associé aux retournements s'il y a au moins cn 1 au niveau du site quand on retourne les molécules en conséquence, pour un petit paramètre constant c.

---

*Un polynôme $P(x)$ de degré m est dit réciproque si, pour tout réel x non nul, on a : $P(\frac{1}{x}) = \frac{p(x)}{x^m}$.

Le maniement de données réelles de cartographie optique est considérablement plus difficile que le problème BFC. Des algorithmes efficaces pour le problème de cartographie optique ont été développés par Anantharaman *et al.*, 1997 [8], Karp et Shamir, 1998 [190] et Lee *et al.*, 1998 [218].

2.9.2 Cartographie de la digestion partielle sondée

Une autre technique utilisée pour trouver une carte physique mène au *problème de la digestion partielle sondée (PPDP)*. Dans cette méthode, l'ADN est partiellement digéré avec une enzyme de restriction, créant ainsi une collection de fragments d'ADN entre toute paire de sites de coupure. Ensuite, une sonde marquée, qui attache l'ADN entre deux sites de coupure, s'hybride à l'ADN partiellement digéré et on mesure les tailles des fragments auxquels la sonde s'hybride. Le problème est de reconstruire les positions des sites à partir du multi-ensemble des longueurs mesurées.

Dans le problème PPDP, le multi-ensemble $X \subset [-s, t]$ est divisé en deux sous-ensembles $X = A \cup B$, avec $A \subset [-s, 0]$ et $B \subset [0, t]$, qui correspondent aux sites de restriction situés à gauche et à droite de la sonde. L'expérimentation PPDP fournit le multi-ensemble $E = \{b - a \ : \ a \in A, b \in B\}$. Le problème est alors de trouver X, étant donné E. Newberg et Naor, 1993 [252] ont montré que le nombre de solutions PPDP peut croître rapidement, au moins aussi vite que $n^{1,72}$.

Chapitre 3

Assemblage de cartes

3.1 Introduction

Pour bien comprendre le problème de l'assemblage de cartes, on peut l'illustrer de la manière suivante. Imaginons plusieurs copies d'un livre qui auraient été découpées avec des ciseaux en quelques milliers de morceaux. Chaque exemplaire est coupé d'une façon qui lui est propre, si bien qu'un morceau d'une copie peut chevaucher une morceau d'une autre copie. Pour chaque morceau et chaque mot tiré d'une liste de mots-clés, on nous dit si le morceau en question contient le mot-clé. À partir de cette information, on souhaite déterminer la table des chevauchements des différents morceaux.

Les techniques de digestions double et partielle permettent au biologiste de construire des cartes (physiques) de restriction de petites molécules d'ADN, comme de l'ADN viral, chloroplaste ou mitochondrial. Cependant, ces méthodes ne fonctionnent pas (ni expérimentalement ni informatiquement) pour de grandes molécules d'ADN. Bien que la première carte de restriction d'un génome viral ait été construite en 1973, il a fallu plus d'une décennie pour construire les premières cartes physiques d'un génome bactérien par l'assemblage de cartes de restriction de petits fragments. Pour étudier une grande molécule d'ADN, les biologistes la cassent en morceaux plus petits, cartographient ou prennent les empreintes de chaque morceau, puis assemblent ces morceaux pour déterminer la carte de la molécule entière. Cette stratégie de cartographie a été développée à l'origine par Olson *et al.*, 1986 [257] pour la levure et par Coulson *et al.*, 1986 [76] pour le nématode. Cependant, la première carte physique de grande échelle a été construite par Kohara *et al.*, 1987 [204] pour la bactérie *E. Coli.*

En général, on commence une cartographie en cassant une molécule d'ADN en petits morceaux à l'aide d'enzymes de restriction. Pour étudier les morceaux individuellement, les biologistes obtiennent de nombreuses copies identiques de chacun d'eux en les *clonant*. Le clonage incorpore un fragment d'ADN dans un *vecteur de clonage* (il s'agit d'une petite molécule d'ADN construite artifi-

ciellement qui provient d'un virus ou d'un autre organisme). Les vecteurs de clonage avec des insertions d'ADN sont introduits dans un hôte bactérien auto-réplicant. Le procédé d'auto-réplication crée alors un nombre considérable de copies de ce fragment, permettant ainsi à sa structure d'être étudiée. Un fragment reproduit de cette façon est appelé un *clone*.

Les biologistes obtiennent ainsi une *banque de clones* constituée de milliers de clones (chacun d'eux représentant un court fragment d'ADN) de la même molécule d'ADN. Les clones de la banque peuvent se chevaucher (le chevauchement peut être obtenu en utilisant plusieurs enzymes de restriction). Une fois la banque de clones établie, les biologistes souhaitent *ordonner* les clones, c'est-à-dire reconstruire le placement relatif des clones le long de la molécule d'ADN. Cette information est perdue lors de la construction de la banque de clones et le processus de reconstruction commence avec la *prise d'empreintes digitales* (ou *fingerprint*) des clones. L'idée est de décrire chaque clone à l'aide d'une empreinte facilement déterminée, qui peut être vue comme un ensemble de « mots-clés » présents dans un clone. Si deux clones présentent des chevauchements importants, leurs empreintes doivent être similaires. S'il est peu probable que des clones ne se chevauchant pas aient des empreintes similaires, alors la prise d'empreintes permet à un biologiste de distinguer les clones qui se chevauchent de ceux qui ne se chevauchent pas, puis de reconstruire l'ordonnancement des clones. Les différents types d'empreintes suivants ont été utilisés dans de nombreux projets de cartographie.

- *Cartes de restriction.* La carte de restriction d'un clone fournit une liste ordonnée de fragments de restriction. Si deux clones ont des cartes de restriction avec quelques fragments consécutifs en commun, il y a des chances qu'ils se chevauchent. Avec cette stratégie, Kohara *et al.*, 1987 [204] ont construit une carte physique du génome *E. Coli.*

- *Tailles des fragments de restriction.* Les tailles des fragments de restriction sont obtenues en coupant un clone avec une enzyme de restriction et en mesurant les tailles des fragments qui en résultent. Ceci est plus simple que de construire une carte de restriction. Bien qu'une liste non ordonnée de tailles de fragments contienne moins d'information qu'une liste ordonnée, elle fournit encore une empreinte adéquate. Ce type d'empreinte a été utilisé par Olson *et al.*, 1986 [257] dans le projet de cartographie de la levure.

- *Données d'hybridation.* Dans cette approche, un clone est exposé à un certain nombre de sondes et on détermine celle qui s'hybride au clone. Les sondes peuvent être de courtes séquences aléatoires ou presque n'importe quel morceau d'ADN préalablement identifié. Le *site du morceau de séquence* (STS pour *Sequence Tag Site*) est un type de sonde particulièrement utile. Les STS sont extraits du brin d'ADN lui-même, souvent à partir des extrémités des clones. Chaque STS est suffisamment long pour

avoir peu de chance d'apparaître une seconde fois dans le brin d'ADN ; par conséquent, il détermine un site unique le long du brin d'ADN. Grâce à la cartographie STS, Chumakov *et al.*, 1992 [68] et Foote *et al.*, 1992 [111] ont construit la première carte physique du génome humain.

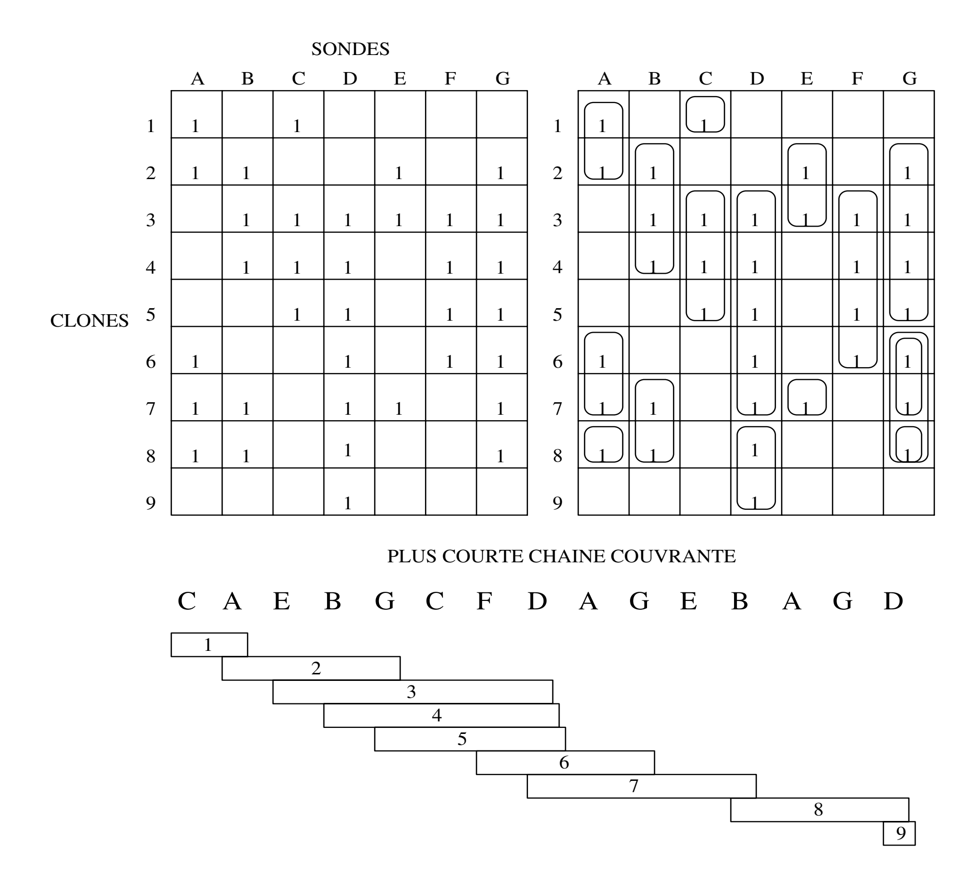


Figure 3.1 – Données d'hybridation et plus courte chaîne couvrante.

La technique STS aboutit à la *cartographie avec des sondes uniques.* Si les sondes sont de courtes séquences aléatoires, elles peuvent s'hybrider à l'ADN en de nombreuses positions, aboutissant ainsi à la *cartographie avec des sondes non uniques.* Pour le problème d'assemblage de cartes avec n clones et m sondes, les données expérimentales consistent en une matrice $D = (d_{ij})$ de taille $n \times m$, où d_{ij} vaut 1 si le clone C_i contient la sonde p_j et 0 sinon (voir figure 3.1). Il faut noter que les données n'indiquent pas combien de fois une sonde apparaît sur un certain clone et qu'elles ne donnent pas non plus l'ordre des sondes le long des clones. Une chaîne S *recouvre* un clone C s'il existe une sous-chaîne de S contenant exactement le même ensemble de sondes que C (on ignore l'ordre

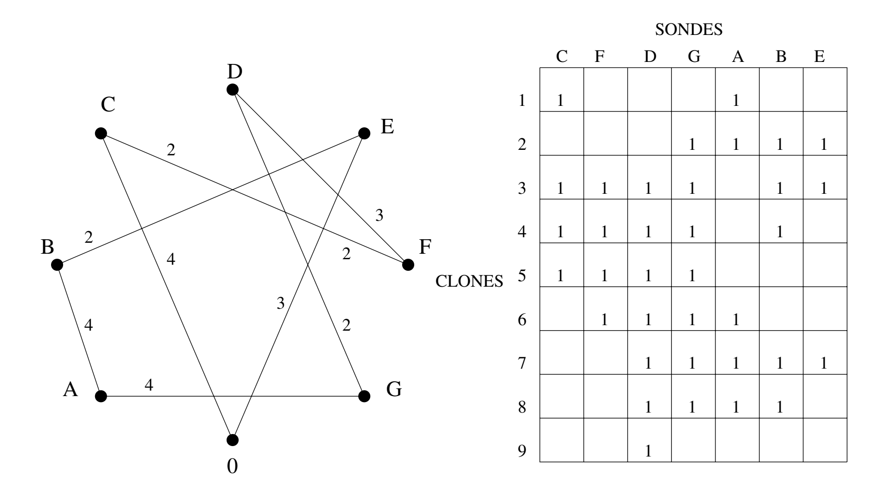
et les multiplicités des sondes dans la sous-chaîne). La chaîne de la figure 3.1 recouvre chacun des neuf clones correspondant à la matrice d'hybridation D. Le problème de la plus courte chaîne couvrante consiste à trouver une plus courte chaîne dans l'alphabet des sondes qui recouvre tous les clones.

Le problème de la plus courte chaîne couvrante est NP-complet. Cependant, si l'ordre des clones est fixé, il peut être résolu en temps polynomial. Alizadeh *et al.*, 1995 [3] ont suggéré une stratégie d'amélioration locale pour le problème de la plus courte chaîne couvrante, fondée sur la découverte de l'intercalation optimale pour un ordre de clone fixé.

Étant donné un ensemble d'intervalles situés sur une même droite, on peut former le *graphe d'intervalles* de celui-ci en associant à chaque intervalle un sommet du graphe et en reliant deux sommets par une arête si les intervalles correspondants se chevauchent (voir figure 3.3). Dans le cas de sondes uniques, chaque matrice d'hybridation sans erreur définit un graphe d'intervalles sur l'ensemble des sommets des clones, dans lequel les clones i et j sont reliés par une arête s'ils ont une sonde en commun. L'étude des graphes d'intervalles a été lancée par Benzer, qui a obtenu des données sur les chevauchements entre les paires de fragments de l'ADN du bactériophage T4. Il a réussi à arranger les données de chevauchement d'une manière qui caractérise la nature linéaire du gène. Le problème de Benzer peut être formulé comme suit : étant donnée l'information qui dit si deux fragments d'un génome se chevauchent ou non, les données sont-elles compatibles avec l'hypothèse selon laquelle les gènes sont disposés dans un ordre linéaire ? Ceci revient à se demander si le graphe de couverture est un graphe d'intervalles.

Les graphes d'intervalles sont étroitement liés aux matrices possédant la propriété des 1 consécutifs. Une matrice $(0,1)$ possède la propriété des *1 consécutifs* si l'on peut permuter ses colonnes de telle sorte que les 1 de chaque ligne apparaissent à des positions consécutives. Dans le cas de sondes uniques, toute matrice d'hybridation sans erreur possède la propriété des 1 consécutifs (la permutation requise sur les colonnes correspond à l'ordonnancement des sondes de gauche à droite). Étant donnée une matrice arbitraire, on s'intéresse à un algorithme qui teste si elle présente la propriété des 1 consécutifs. La caractérisation des graphes d'intervalles et des matrices ayant la propriété des 1 consécutifs a été donnée par Gilmore et Hoffman, 1964 [128] et par Fulkerson et Gross, 1965 [114]. Booth et Leuker, 1976 [40] ont développé une structure de données appelée *arbre-PQ* qui conduit à un algorithme de complexité linéaire qui reconnaît la propriété des 1 consécutifs. Étant donnée une matrice d'hybridation sans erreur, l'algorithme de Booth-Leuker construit une représentation compacte de tous les ordonnancements corrects des sondes en un temps linéaire. Cependant, leur approche ne tolère aucune erreur expérimentale. Alizadeh *et al.*, 1995 [3] ont imaginé une procédure alternative simple pour l'ordonnancement des sondes. En présence d'erreurs expérimentales, cette procédure peut échouer mais, dans la plupart des cas, elle demeure une bonne heuristique pour la construction d'un ordonnancement de sondes.

L'erreur d'hybridation la plus fréquente est un *faux négatif*, où l'incidence entre une sonde et un clone a lieu mais n'est pas observée. De surcroît, les données d'hybridation sont sujettes à des faux positifs et à des erreurs dues aux *anomalies clonales*. Les technologies de clonage souffrent de différentes anomalies clonales. Les premières banques de clones étaient fondées sur les vecteurs du bactériophage λ et recevaient jusqu'à 25 kb d'ADN. Les *cosmides* représentent un autre vecteur qui combine les séquences d'ADN à partir de *plasmides* et d'une région du génome λ. Avec les cosmides, la plus grande taille d'une incrustation d'ADN est de 45 kb et il faudrait 70 000 cosmides pour couvrir le génome humain. Pour réduire ce nombre, on a développé les *chromosomes artificiels de levure* (YAC pour *Yeast Artificial Chromosomes*) pour cloner des fragments d'ADN plus longs (jusqu'à 1 000 kb). Bien que les YAC aient été utilisés dans de nombreux projets de cartographie, il existe un certain nombre d'anomalies qui leur sont associées ; la plus courante est le *chimérisme*. Un clone chimère est constitué de deux segments d'ADN distincts reliés par erreur lors d'un processus de clonage. On estime que 10% à 60% des clones des banques de YAC sont des chimères. Les chimères peuvent survenir par *coligation* de différents fragments d'ADN ou par *recombinaison* de deux molécules d'ADN. Un autre problème avec les YAC est que de nombreux clones sont instables et ont tendance à supprimer des régions internes. Comparés aux YAC, les systèmes de clonage BAC (*chromosome artificiel bactérien*), fondés sur le génome *E. Coli*, réduisent de façon significative le problème du chimérisme.



CLONES	C	F	D	G	A	B	E
1	1				1		
2				1	1	1	1
3	1	1	1	1		1	1
4	1	1	1	1		1	
5	1	1	1	1			
6		1	1	1	1		
7			1	1	1	1	1
8			1	1	1	1	
9			1				

Figure 3.2 – Le cycle le plus court pour le TSP de la distance de Hamming correspondant à la matrice d'hybridation de la figure 3.1, avec un ordre des colonnes différent. Le cycle minimal exhibé définit l'ordonnancement des clones ayant le nombre minimal de brèches. Dans cet ordonnancement, les clones 1, 3 et 4 sont des chimères.

Un bon ordonnancement des sondes correspond à une permutation π des colonnes de la matrice d'hybridation D conduisant à une matrice D_π. Chaque

ligne de D_π qui correspond à un clone normal contient un bloc de 1, tandis que toute ligne qui correspond à un clone chimère en contient deux. On définit une *brèche* comme un bloc de zéros sur une ligne encadré par des 1. Le nombre de brèches dans D_π est égal au nombre de clones chimères. Une erreur de faux négatif coupe typiquement un bloc de 1 en deux parties, créant ainsi une brèche. Une erreur de faux positif, quant à elle, sépare une brèche en deux brèches. Par conséquent, le nombre de brèches dans D_π est approximativement égal au nombre de clones chimères, auquel on ajoute le nombre d'erreurs d'hybridation. Ceci suggère un principe heuristique selon lequel une permutation des colonnes qui minimise le nombre de brèches correspondrait à un bon ordonnancement de sondes (Alizadeh *et al.*, 1995 [3], Greenberg et Istrail, 1995 [137]). Minimiser le nombre de brèches peut être vu comme un cas particulier du *problème du voyageur de commerce* (TSP pour *Traveling Salesman Problem*), que l'on appelle le *TSP de la distance de Hamming.* Dans ce problème, les villes sont les colonnes de D, avec une colonne supplémentaire entièrement constituée de zéros et la distance entre deux villes est la distance de Hamming entre les colonnes correspondantes, c'est-à-dire le nombre de positions dans lesquelles les deux colonnes diffèrent (voir figure 3.2).

3.2 Cartographie avec sondes non uniques

Poustka *et al.*, 1986 [279] ont proposé la cartographie physique utilisant les empreintes d'hybridation avec de courtes sondes. L'avantage de cette approche est que la production de sondes est bon marché et directe. Cependant, le taux d'erreurs lors d'expériences d'hybridation avec des sondes courtes est très élevé ; par conséquent, très peu de projets de cartographie utilisant des sondes non uniques ont abouti (Hoheisel *et al.*, 1993 [165]). Les erreurs d'hybridation, ainsi que l'absence de bons algorithmes permettant la construction de cartes ont été les obstacles majeurs à l'utilisation à grande échelle de cette méthode. Récemment, Alizadeh *et al.*, 1995 [3] et Mayraz et Shamir, 1999 [234] ont développé des algorithmes qui fonctionnent correctement en présence d'erreurs d'hybridation.

Un *placement* est l'association d'un intervalle du segment $[0, N]$ à chaque clone (l'intervalle $[0, N]$ correspond à la molécule d'ADN entière). Une *intercalation* correspond à une spécification de l'ordonnancement linéaire des $2n$ extrémités de n intervalles. Une intercalation peut être vue comme une classe d'équivalence de placements ayant une structure topologique commune. Étant donnée une matrice D, le problème d'assemblage de cartes consiste à déterminer l'intercalation la plus vraisemblable (Alizadeh *et al.*, 1995 [3]).

Alizadeh *et al.*, 1995 [3] ont donné une signification précise à la notion de « plus grande vraisemblance » pour le modèle stochastique de Lander et Waterman, 1988 [214] ; ils ont montré qu'elle correspond à la plus courte chaîne couvrante. Dans ce modèle, les clones de même longueur sont placés à des positions indépendantes aléatoires le long de l'ADN et les sondes sont placées

le long de l'ADN selon des processus de Poisson mutuellement indépendants. Alizadeh *et al.*, 1995 [3] ont imaginé une fonction de probabilité maximale pour ce modèle. L'approximation la plus simple de cette fonction consiste à trouver une intercalation qui minimise le nombre d'occurrences de sondes nécessaires pour expliquer les données d'hybridation, le problème de la plus courte chaîne couvrante.

Dans la suite, on suppose qu'aucun clone n'en contient un autre. Une chaîne S *recouvre une permutation de clones* π si elle recouvre tous les clones dans l'ordre donné par π. Par exemple, la chaîne $ABACBACDBCE$ recouvre la permutation $(3,2,4,1)$ des clones C_1, C_2, C_3, C_4 qui s'hybrident aux sondes A, B, C, D, E suivantes :

$$C_1 - \{B, C, E\},\ C_2 - \{A, B, C, D\},\ C_3 - \{A, B, C\},\ C_4 - \{B, C, D\}.$$

Soit $c(\pi)$ la longueur de la plus courte chaîne recouvrant π. La figure 3.1 présente l'une des plus petites chaînes couvrantes pour la permutation $\pi = (1, 2, \ldots, 9)$, de longueur $c(\pi) = 15$. Alizadeh *et al.*, 1995 [3] ont imaginé un algorithme polynomial permettant de trouver $c(\pi)$, ainsi qu'un algorithme d'amélioration pour approximer $\min_\pi c(\pi)$.

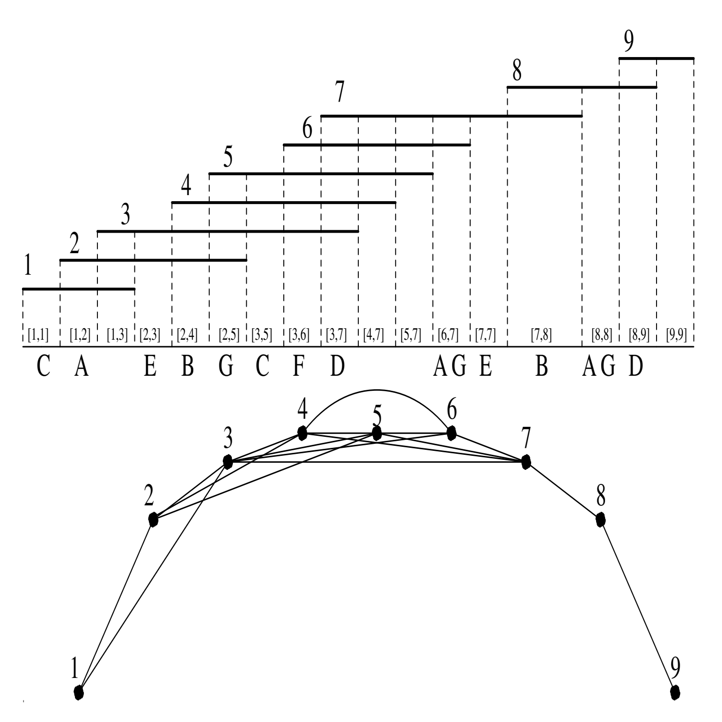


Figure 3.3 – Intervalles atomiques.

Les extrémités des clones partitionnent le segment en un certain nombre d'*intervalles atomiques*, où un intervalle atomique est un intervalle maximal

qui ne contient aucune extrémité de clone (voir figure 3.3). Chaque intervalle atomique est contenu dans les clones $i, \ldots, j$, où i (resp. j) représente le clone situé à l'extrême gauche (resp. droite) qui contient l'intervalle atomique. Un tel intervalle atomique est noté $[i, j]$. Les intervalles $[i, j]$ et $[i', j']$ sont *en conflit* si l'on a $i < i' < j' < j$. Notons que chaque ensemble d'intervalles atomiques est *sans conflit*, c'est-à-dire qu'il ne contient pas d'intervalles en conflit (encore une fois, on suppose qu'aucun clone n'en contient un autre).

Un ensemble d'intervalles $\mathcal{I} = ([i, j])$ est dit *compatible* s'il existe une intercalation de clones pour laquelle tout intervalle de $\mathcal{I}$ est atomique.

Lemme 3.1 *Un ensemble d'intervalles est compatible si et seulement s'il est sans conflit.*

Preuve Par récurrence sur le nombre de clones. ∎

Pour les données présentées en figure 3.1, les clones contenant une sonde A sont organisés en *deux* intervalles : $[1, 2]$ et $[6, 8]$. Toute sonde A dans une chaîne couvrante définit un intervalle atomique $[i, j]$ dans l'intercalation de clones correspondante ; elle engendre donc *un* intervalle de longueur $j - i + 1$ dans la colonne A de la matrice d'hybridation. Par conséquent, A apparaît au moins deux fois dans chaque chaîne couvrante. Cela nous donne une borne inférieure de $2 + 2 + 2 + 1 + 2 + 1 + 1 = 11$ pour la longueur de la plus courte chaîne couvrante. Cependant, il n'existe pas de chaîne couvrante de longueur 11, car certains intervalles dans la matrice d'hybridation sont en conflit (par exemple, un intervalle $[3, 5]$ pour C est en conflit avec un intervalle $[2, 8]$ pour G). Il est à noter que toute chaîne couvrante de longueur t définit un ensemble compatible de t intervalles. Vice versa, tout ensemble compatible de t intervalles définit une chaîne couvrante de longueur t. Ainsi le lemme 3.1 permet de formuler le problème de la plus courte chaîne couvrante comme suit : subdiviser un ensemble d'intervalles pour une matrice d'hybridation en un *ensemble sans conflit avec un nombre minimum d'intervalles.* Par exemple, une façon d'éviter un conflit entre les intervalles $[3, 5]$ et $[2, 8]$ est de subdiviser l'intervalle $[2, 8]$ en $[2, 5]$ et $[6, 8]$. Cette subdivision laisse encore l'intervalle $[7, 7]$ pour E en conflit avec les intervalles $[6, 8]$ pour A, $[3, 9]$ pour D et $[6, 8]$ pour G. Après avoir subdivisé ces trois intervalles, on aboutit à l'ensemble sans conflit de $11 + 4 = 15$ intervalles (matrice de droite dans la figure 3.1). Ces quinze intervalles engendrent la plus courte chaîne suivante :

$$\underbrace{[1,1]}_{C}\,\underbrace{[1,2]}_{A}\,\underbrace{[2,3]}_{E}\,\underbrace{[2,4]}_{B}\,\underbrace{[2,5]}_{G}\,\underbrace{[3,5]}_{C}\,\underbrace{[3,6]}_{F}\,\underbrace{[3,7]}_{D}\,\underbrace{[6,7]}_{A}\,\underbrace{[6,7]}_{G}\,\underbrace{[7,7]}_{E}\,\underbrace{[7,8]}_{B}\,\underbrace{[8,8]}_{A}\,\underbrace{[8,8]}_{G}\,\underbrace{[8,9]}_{D}$$

Décrivons à présent une approche gloutonne de la subdivision qui produit la plus courte chaîne couvrante. Soient $[i, j]$ et $[i', j']$ deux intervalles en conflit, avec $i < i' \leq j' < j$ et j' minimal parmi tous les intervalles en conflit. Il est clair que $[i, j]$ doit être coupé « avant » j' dans toute subdivision (autrement dit, dans toute subdivision de $[i, j]$, il existe un intervalle $[i, t]$ avec $t \leq j'$). Cette

observation suggère une stratégie de subdivision fondée sur la découpe de tout intervalle « aussi loin que possible » pour éviter les conflits. Plus précisément, pour un intervalle donné $[i, j]$, soit $\mathcal{I}$ l'ensemble des intervalles contenus dans $[i, j]$. Soit t un nombre maximal d'intervalles non chevauchants de $\mathcal{I}$, de sorte que $[i_1, j_1]$ soit un intervalle avec j_1 minimal dans $\mathcal{I}$, que $[i_2, j_2]$ soit un intervalle tel que j_2 soit minimal parmi les intervalles avec $i_2 > j_1$ dans $\mathcal{I}$, ..., et que $[i_t, j_t]$ soit un intervalle tel que j_t soit minimal parmi les intervalles avec $i_t > j_{t-1}$ dans $\mathcal{I}$. À première vue, on a l'impression que la partition de l'intervalle $[i, j]$ en $t+1$ intervalles $[i, j_1], [j_1 + 1, j_2], [j_2 + 1, j_3], \ldots, [j_{t-1} + 1, j_t], [j_t + 1, j]$ aboutit à la solution du problème de la plus courte chaîne couvrante. Malheureusement, ce n'est pas le cas comme le montre un exemple de la figure 3.4 (proposé par Tao Jiang). Cependant, une simple modification du dernier de ces $t + 1$ intervalles mène à la solution du problème (voir la figure 3.4 pour un exemple). Notons que, dans ce cas, la plus courte chaîne couvrante présente un clone avec une double occurrence de la sonde D.

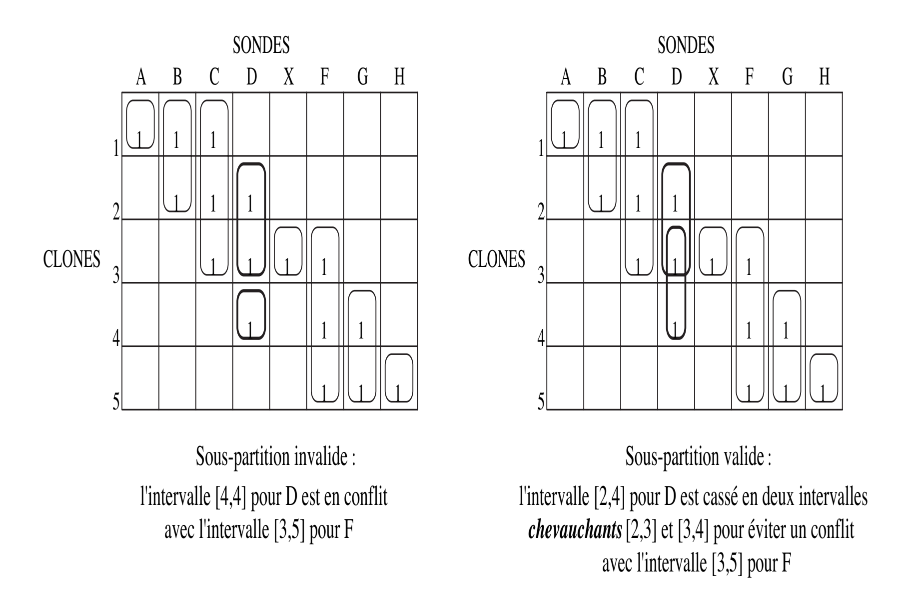


Figure 3.4 – La plus courte chaîne couvrante peut contenir des clones avec des répétitions de sondes (comme CDXDF).

3.3 Cartographie avec sondes uniques

Soit $\{1, \ldots, m\}$ l'ensemble des sondes et soit C_i l'ensemble des clones incidents à la sonde i. Un clone X *contient* un clone Y si l'ensemble des sondes incidentes à X contient (strictement) l'ensemble des sondes incidentes à Y. Par la suite, on suppose que l'ensemble des clones vérifie les conditions suivantes :

- *Non-inclusion.* Aucun clone n'en contient un autre.

- *Connexité.* Pour toute partition de l'ensemble des sondes en deux ensembles non vides A et B, il existe des sondes $i \in A$ et $j \in B$ telles que $C_i \cap C_j$ soit non vide.

- *Unicité.* $C_i \neq C_j$ pour $i \neq j$.

Il n'y a pas de perte de généralité essentielle dans ces hypothèses, puisque tout ensemble de clones peut être filtré en éliminant les clones qui en contiennent d'autres. Le lemme suivant donne une nouvelle formulation de la propriété des 1 consécutifs :

Lemme 3.2 *Soit* $(1, \ldots, m)$ *l'ordonnancement correct des sondes et soient* i, j *et* k *des entiers qui vérifient* $1 \leq i < j < k \leq m$. *Alors, dans le cas où il n'y a pas d'erreur, on a :* $|C_i \cap C_j| \geq |C_i \cap C_k|$ *et* $|C_k \cap C_j| \geq |C_i \cap C_k|$.

Étant donnée une sonde i, comment peut-on trouver des sondes adjacentes $i-1$ et $i+1$ dans l'ordonnancement correct des sondes ? Le lemme 3.2 suggère que ces sondes figurent parmi celles qui maximisent $|C_i \cap C_k|$. Autrement dit, elles doivent vérifier soit $|C_i \cap C_{i-1}| = \max_{k \neq i} |C_i \cap C_k|$, soit $|C_i \cap C_{i+1}| = \max_{k \neq i} |C_i \cap C_k|$. Si $\max_{k \neq i} |C_i \cap C_k|$ n'est obtenu que pour la sonde k, alors soit $k = i-1$, soit $k = i+1$. Si $\max_{k \neq i} |C_i \cap C_k|$ est obtenu pour plusieurs sondes, alors il est facile de voir que l'une de celles avec $|C_k|$ minimal correspond à $i-1$ ou à $i+1$. Ces observations mènent à un algorithme efficace pour l'ordonnancement des sondes qui commence par une sonde i choisie aléatoirement et tente de trouver une sonde adjacente ($i-1$ ou $i+1$) à l'étape suivante. À l'étape $(k+1)$, l'algorithme essaie d'étendre vers la droite ou vers la gauche un bloc de k sondes consécutives déjà trouvé à l'aide du lemme 3.2.

Pour chaque sonde i, on définit la relation d'ordre partiel $\succ_i$ sur l'ensemble des sondes par $j \succ_i k$ si l'on a soit

$$|C_i \cap C_j| > |C_i \cap C_k|$$

soit

$$|C_i \cap C_j| = |C_i \cap C_k| \neq \emptyset \text{ et } |C_j| < |C_k|.$$

Il est clair que, si l'on a $j \succ_i k$, alors la sonde k ne se trouve pas entre la sonde i et la sonde j dans le véritable ordonnancement. En outre, un élément maximal dans $\succ_i$ est soit $i-1$, soit $i+1$.

Soit $N(i)$ l'ensemble des sondes qui apparaissent avec la sonde i sur au moins un clone.

Lemme 3.3 *L'ordonnancement correct des sondes est déterminé de façon unique à une inversion près si les propriétés suivantes sont satisfaites pour tout i :*

- *les sondes dans $N(i)$ apparaissent consécutivement dans l'ordonnancement comme un bloc $B(i)$.*
- *en commençant à la sonde i et en allant vers la droite ou vers la gauche, les sondes dans $B(i)$ forment une chaîne décroissante selon l'ordre partiel $\succ_i$.*

Le lemme motive l'algorithme suivant, qui trouve le véritable ordonnancement des sondes (Alizadeh *et al.*, 1995 [4]). Tout au long de l'algorithme, la variable $\pi = \pi_{premier} \ldots \pi_{dernier}$ représente une séquence de sondes consécutives dans l'ordonnancement correct. Au démarrage, π est une séquence constituée d'une seule sonde. Chaque étape ajoute un élément à π de la façon suivante :

- si chaque élément de $N(\pi_{dernier})$ se trouve dans π, on remplace la séquence π par son inverse.
- on choisit une sonde $k \notin \pi$ qui est la plus grande des sondes de $N(\pi_{dernier})$ selon l'ordre $\succ_{\pi_{dernier}}$. Si l'on a $\pi_{dernier} \succ_k \pi_{premier}$, on ajoute k à la fin de π ; sinon, on l'ajoute au début de π.

L'algorithme s'arrête lorsque π contient toutes les sondes. Le lemme 3.2 implique qu'après chaque étape, les sondes dans π forment un bloc consécutif dans l'ordonnancement correct. Puisqu'un nouvel élément est ajouté à π à chaque étape, l'algorithme donne le bon ordonnancement π après m étapes. Alizadeh *et al.*, 1995 [4] ont établi qu'une variante de cet algorithme fonctionne bien, même en présence d'erreurs d'hybridation et de clones chimères.

3.4 Graphes d'intervalles

Golumbic, 1980 [132] est une excellente introduction aux graphes d'intervalles et notre présentation s'en inspire.

Un *graphe triangulé* est un graphe dans lequel chaque cycle simple de longueur supérieure à trois possède une *corde.*[*] Le graphe de la « maison » dans la figure 3.5 n'est pas triangulé car il contient un 4-cycle sans corde.

Lemme 3.4 *Tout graphe d'intervalles est triangulé.*

[*]Une corde est une arête qui relie deux sommets non consécutifs d'un cycle.

Tout graphe triangulé n'est cependant pas un graphe d'intervalles ; par exemple, le graphe en étoile présenté en figure 3.5 n'est pas un graphe d'intervalles (prouvez-le !). Un graphe non orienté possède la *propriété de l'orientation transitive* si l'on peut associer à chaque arête une direction, de sorte que le graphe orienté $G(V, E)$[†] obtenu vérifie la condition suivante pour tout triplet de sommets (a, b, c) : $(a, b) \in E$ et $(b, c) \in E$ impliquent $(a, c) \in E$. Un graphe non orienté qui est transitivement orientable est appelé un graphe de *comparaison* (voir figure 3.5). Le complémentaire[‡] du graphe en étoile présenté en figure 3.5 n'est pas un graphe de comparaison.

Lemme 3.5 *Le complémentaire d'un graphe d'intervalles est un graphe de comparaison.*

Preuve Pour des intervalles non chevauchants $[i, j]$ et $[i', j']$ avec $j \leq i'$, on dirige l'arête correspondante dans le complémentaire du graphe d'intervalles dans le sens $[i, j] \to [i', j']$. ■

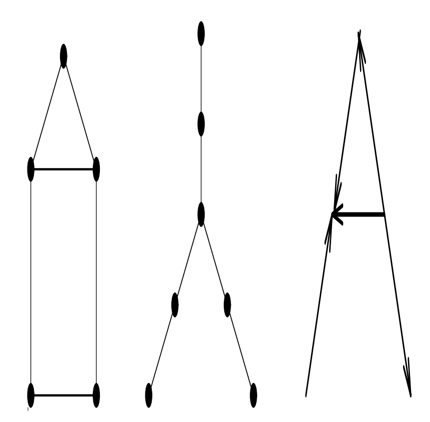


Figure 3.5 – (i) Le graphe de la maison n'est pas un graphe d'intervalles car il n'est pas triangulé. (ii) Le graphe en étoile n'est pas un graphe d'intervalles car son complémentaire n'est pas un graphe de comparaison. (iii) Une orientation transitive d'un graphe « A ».

Un graphe *complet* est un graphe dans lequel toute paire de sommets forme une arête. Une clique d'un graphe G est un sous-graphe complet de G. Une clique est maximale si elle n'est contenue dans aucune autre clique. Le graphe présenté en figure 3.3 possède cinq cliques maximales : $\{1, 2, 3\}$, $\{2, 3, 4, 5\}$, $\{3, 4, 5, 6, 7\}$, $\{7, 8\}$ et $\{8, 9\}$. Le théorème suivant établit la caractérisation des graphes d'intervalles :

[†] La notation $G(V, E)$ désigne un graphe G d'ensemble de sommets V et d'ensemble d'arêtes E.

[‡] Le *complémentaire* $\overline{G}$ d'un graphe $G = (V, E)$ est le graphe d'ensemble de sommets V et d'ensemble d'arêtes $\overline{E} = \binom{V}{2} \setminus E$.

Théorème 3.1 *(Gilmore et Hoffman, 1964 [128]) Soit G un graphe non orienté. Les assertions suivantes sont équivalentes :*
(i) G est un graphe d'intervalles.
(ii) G est un graphe triangulé et son complémentaire est un graphe de comparaison.
(iii) Les cliques maximales de G peuvent être ordonnées linéairement, de sorte que, pour chaque sommet x de G, les cliques maximales contenant x apparaissent de façon consécutive.

Preuve (i) $\Rightarrow$ (ii). C'est une conséquence des lemmes 3.4 et 3.5.

(ii) $\Rightarrow$ (iii). Soit $G(V, E)$ un graphe triangulé et soit F une orientation transitive du complémentaire $\overline{G}(V, \overline{E})$. Soient A_1 et A_2 des cliques maximales de G. Il est clair qu'il existe une arête dans F ayant une extrémité dans A_1 et l'autre dans A_2 (sinon, $A_1 \cup A_2$ formerait une clique de G). Il est facile de voir que toutes les arêtes de $\overline{E}$ qui relient A_1 et A_2 ont la même orientation. (Indication : si les arêtes (a_1, a_2) et (a'_1, a'_2) relient A_1 et A_2 comme dans la figure 3.6 (à gauche), alors au moins une des arêtes (a_1, a'_2) et (a'_1, a_2) appartient à $\overline{E}$. De quelle façon est-elle orientée ?). On ordonne les cliques maximales selon l'orientation des arêtes dans F : on a $A_1 < A_2$ si et seulement s'il existe une arête de F qui relie A_1 et A_2 en étant dirigée vers A_2. Montrons à présent que cet ordre est transitif et que, par conséquent, il définit un ordre total[§] des cliques.

Supposons que l'on ait $A_1 < A_2$ et $A_2 < A_3$. Il existe alors des arêtes (a_1, a'_2) et (a''_2, a_3) dans F avec $a_1 \in A_1$, $a'_2, a''_2 \in A_2$ et $a_3 \in A_3$ (voir figure 3.6 (à droite)). Si l'on a soit $(a'_2, a_3) \notin E$, soit $(a_1, a''_2) \notin E$, alors on obtient : $(a_1, a_3) \in F$ et $A_1 < A_3$. Par conséquent, supposons que les arêtes $(a_1, a''_2), (a''_2, a'_2)$ et (a'_2, a_3) soient toutes dans E. Comme G ne contient pas de 4-cycle sans corde, on en déduit que (a_1, a_3) n'est pas une arête de E et la transitivité de l'ordre sur F implique que (a_1, a_3) est dans F. D'où $A_1 < A_3$, ce qui prouve la transitivité de l'ordre.

Soit $A_1, \ldots, A_m$ l'ordonnancement linéaire des cliques maximales. Supposons qu'il existe des cliques $A_i < A_j < A_k$ avec $x \in A_i$, $x \notin A_j$ et $x \in A_k$. Comme on sait que $x \notin A_j$, il y a un sommet $y \in A_j$ tel que l'on ait $(x, y) \notin E$. Mais $A_i < A_j$ implique $(x, y) \in F$, tandis que $A_j < A_k$ implique $(y, x) \in F$, d'où la contradiction.

(iii) $\Rightarrow$ (i). Pour chaque sommet x, soit $I(x)$ l'ensemble de toutes les cliques maximales de G qui contiennent x. Les ensembles $I(x)$, pour $x \in V$, forment les intervalles du graphe d'intervalles G. ■

Le théorème 3.1 ramène la reconnaissance d'un graphe d'intervalles à la reconnaissance de graphes triangulés et de comparaison (Fulkerson et Gross, 1965 [114], Pnueli *et al.*, 1971 [277]).

[§]Une *relation d'ordre total* est une relation d'ordre pour laquelle deux éléments quelconques de l'ensemble considéré peuvent toujours être comparés ; en d'autres termes, pour tout couple d'éléments (x, y), on a soit $x \leq y$, soit $y \leq x$.

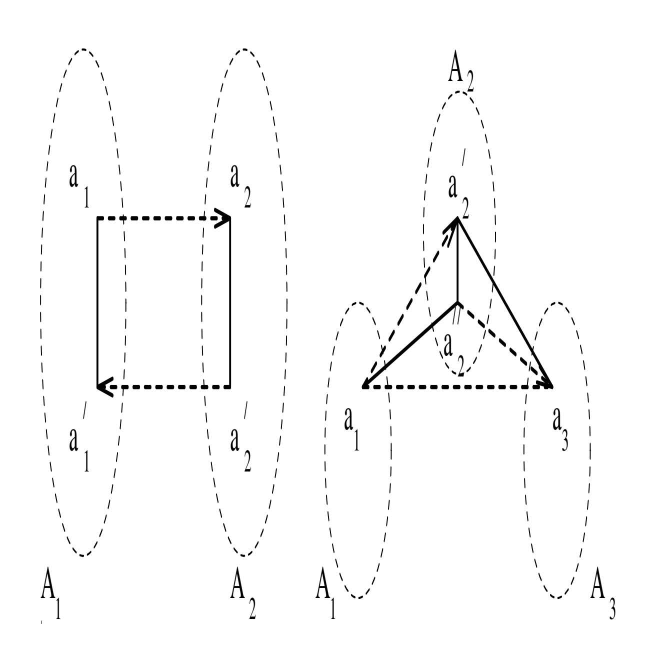


Figure 3.6 – Illustrations liées à la preuve du théorème 3.1.

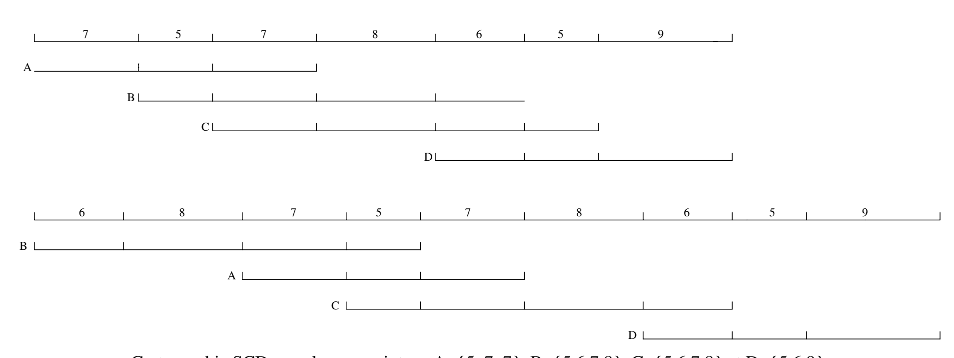
3.5 Cartographie avec empreintes de fragments de restriction

Le cas le plus simple de cartographie avec empreintes de fragments de restriction est la cartographie de la *digestion complète simple* (SCD pour *Single Complete Digest*) (Olson *et al.*, 1986 [257]). Dans ce cas, l'empreinte d'un clone est un *multi-ensemble* formé des tailles de ses fragments de restriction dans une digestion par une enzyme de restriction. Une carte SCD (voir figure 3.7) est un placement de clones et de fragments de restriction compatible avec les données SCD (Gillett *et al.*, 1995 [127]).

Problème de la cartographie SCD Trouver la carte minimale (c'est-à-dire une carte présentant le nombre minimal de fragments de restriction) qui soit compatible avec les données SCD.

Le problème consistant à trouver la carte la plus compacte est NP-complet. En pratique, les empreintes des clones fournissent souvent une évidence statistique forte qui permet d'estimer l'ordonnancement des clones. Jiang et Karp, 1998 [179] ont étudié le problème consistant à trouver la carte la plus compacte pour des clones avec un ordonnancement donné.

Supposons qu'aucun clone n'en contienne un autre et que tout clone commence et finisse avec le site de l'enzyme de restriction. Jiang et Karp ont formulé la cartographie SCD avec un ordonnancement connu des clones sous forme d'un



Cartographie SCD avec les empreintes : A={5, 7, 7}, B={5,6,7,8}, C={5,6,7,8} et D={5,6,9}.
La carte la plus compacte a un cardinal ègal á 7 (en haut) ; une autre carte possible a pour cardinal 9 (en bas).

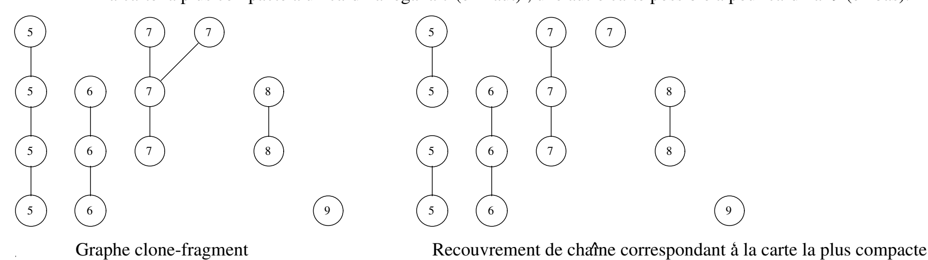


Figure 3.7 – Cartographie SCD.

problème de *recouvrement de chemin forcé* sur un graphe spécial à plusieurs étages. Soit $\mathcal{S} = \{S_1, \ldots, S_n\}$ un exemple de cartographie SCD, où chaque S_i est un multi-ensemble représentant l'empreinte du i-ième clone dans l'ordonnancement de clones par les extrémités gauches. Un graphe *étiqueté à étages multiples* G (appelé graphe *clone-fragment*) est constitué de n étages, le i-ième étage contenant $|S_i|$ sommets. À l'étage i, G possède un sommet pour chaque élément x de S_i (doubles inclus), avec l'étiquette x. Deux sommets sont reliés s'ils sont à des étages adjacents et s'ils ont des étiquettes identiques (voir figure 3.7).

Intuitivement, un chemin dans G détermine un ensemble de fragments de restriction sur des clones consécutifs qui peuvent être placés au même endroit dans une carte. Un chemin passant par les étages $i, \ldots, j$ est simplement noté $[i, j]$.

Un *recouvrement de chemin* est une collection de chemins telle que chaque sommet est contenu dans exactement un chemin. Il est clair que toute carte pour $\mathcal{S}$ correspond à un recouvrement de chemin de G de même cardinal (la figure 3.7 présente un recouvrement de chemin de cardinal 7). En revanche, la réciproque n'est pas vraie ; par exemple, le graphe clone-fragment présenté en figure 3.7 a un recouvrement de chemin de cardinal 6 qui ne correspond à aucune carte. Ceci est dû au fait que certains chemins sont en conflit (à comparer au lemme 3.1). Les chemins $[i, j]$ et $[i', j']$ sont *en conflit* si l'on a $i < i' < j' < j$.

Un recouvrement de chemin est sans conflit s'il n'a pas de chemins en conflit. Un recouvrement de chemin de G est *compatible* s'il correspond à une carte de $\mathcal{S}$ de même cardinal. Comme pour le lemme 3.1, on a :

Lemme 3.6 *Un recouvrement de chemin est compatible si et seulement s'il est sans conflit.*

Construire la carte la plus compacte de $\mathcal{S}$ équivaut donc à trouver le plus petit recouvrement de chemin sans conflit de G. Bien que le problème consistant à trouver le plus petit recouvrement de chemin sans conflit de G soit semblable au problème de la plus courte chaîne couvrante, leur complexité calculatoire est très différente. Jiang et Karp, 1998 [179] ont décrit un algorithme de rapport d'approximation 2 pour la cartographie SCD avec un ordonnancement de clones donné.

3.6 Quelques autres problèmes et approches

3.6.1 Statistique de Lander-Waterman

Lorsqu'ils débutent un projet de cartographie physique, les biologistes doivent décider du nombre de clones nécessaires pour construire une carte. Au minimum, il faut que la quasi-totalité du génome soit recouverte par des clones. Dans l'un des premiers projets de cartographie, Olson *et al.*, 1986 [257] ont construit une banque contenant $n = 4\,946$ clones. Chaque clone portait un fragment d'ADN d'une longueur moyenne de $L = 15\,000$ nucléotides. Cette banque de clones représentait une molécule d'ADN de longueur $G = 2 \times 10^7$; autrement dit, chaque nucléotide était représenté dans $\frac{NL}{G} \approx 4$ clones en moyenne. Le nombre $c = \frac{NL}{G}$ est appelé la *couverture* d'une banque de clones. Une banque typique fournit une couverture située dans un intervalle allant de 5 à 10. Quand ce projet a commencé, on ne savait pas exactement quel pourcentage du génome de la levure serait recouvert par 4946 clones.

Lander et Waterman, 1988 [214] ont étudié un modèle probabiliste dans lequel les clones de même longueur ont été dispersés en des positions aléatoires indépendantes le long de l'ADN. On définit une brèche comme étant un intervalle qui n'est recouvert par aucun clone ; un contig est un intervalle maximal sans brèche. Lander et Waterman, 1988 [214] ont démontré que le nombre attendu de brèches est d'environ ne^{-c} et que la fraction attendue d'ADN non recouvert par un clone est d'approximativement e^{-c}. Le projet de cartographie d'Olson *et al.*, 1986 [257] a abouti à 1 422 contigs, ce qui est très proche de l'estimation de Lander-Waterman de 1 457 contigs. Arratia *et al.*, 1991 [11] sont allés plus loin dans le développement des statistiques de Lander-Waterman dans le cas d'empreintes d'hybridation.

3.6.2 Criblage de banques de clones

Une approche naïve pour obtenir une matrice d'hybridation pour n clones et m sondes nécessite $n \times m$ expériences d'hybridation. Le but du *regroupement* est de réduire ce nombre. Si le nombre de 1 dans une matrice d'hybridation est petit (c'est le cas de la cartographie avec des sondes uniques), alors la plupart des expériences donnent des résultats négatifs. Il est clair que regrouper des clones en groupes et tester les sondes sur ceux-ci peut économiser des efforts expérimentaux.

Pour plus de simplicité, supposons que $\sqrt{n}$ soit un entier et imaginons n clones comme étant des éléments d'un tableau de taille $\sqrt{n} \times \sqrt{n}$. On regroupe les clones correspondant à chaque ligne et chaque colonne de ce tableau. L'ensemble qui en résulte est constitué de $2\sqrt{n}$ groupes, chacun contenant $\sqrt{n}$ clones. Ceci réduit les efforts expérimentaux de $n \times m$ à $2\sqrt{n} \times m$ hybridations, mais complique le problème calculatoire d'assemblage de carte (Evans et Lewis, 1989 [98] et Barillot *et al.*, 1991 [25]). Chumakov *et al.*, 1992 [68] ont utilisé cette stratégie de regroupement pour la construction de la première carte physique humaine. L'analyse informatique des stratégies de regroupement est liée au problème suivant :

Problème du test de groupe Trouver les membres distingués d'un ensemble d'objets $\mathcal{L}$ en posant le nombre minimum de questions de la forme : « l'ensemble $Q \subset \mathcal{L}$ contient-il un objet distingué ? ».

Poser les questions correspond à tester le groupe avec une sonde. Pour les applications de cartographie, il est plus efficace au niveau du coût de poser les questions en parallèle (test de groupe non adaptatif). Bruno *et al.*, 1995 [51] ont préconisé une stratégie de regroupement selon un modèle de k-ensemble aléatoire qui possède certains avantages comparé au modèle de regroupement en lignes et colonnes. Dans le modèle de k-ensemble aléatoire, chaque clone apparaît dans k groupes, les choix des k groupes étant équiprobables. On peut consulter Knill *et al.*, 1998 [200] pour l'analyse du test de groupe non adaptatif en présence d'erreurs.

3.6.3 Cartographie par hybrides d'irradiation

La cartographie par hybrides d'irradiation (RH pour *Radiation hybrid*) (Cox *et al.*, 1990 [77]) est une stratégie expérimentale utilisant un regroupement aléatoire qui a lieu dans la nature. La cartographie RH implique l'exposition de cellules humaines à l'irradiation, ce qui casse chaque chromosome en des fragments aléatoires. Ces fragments sont ensuite « sauvés » en fusionnant les cellules humaines avec des cellules de hamster qui incorporent un sous-ensemble aléatoire des fragments d'ADN humain dans leurs chromosomes. On peut imaginer les fragments humains comme étant des clones ; les cellules de hamster sont alors des groupes constitués de ces clones. La cellule hybride qui en résulte

peut devenir une lignée cellulaire contenant un groupe du fragment du génome humain. La figure 3.8 présente une expérience de cartographie RH avec trois lignées cellulaires hybrides et quatre sondes (marqueurs) qui aboutit à une *matrice de criblage hybride* de taille 4×3. Le problème de la cartographie par hybrides d'irradiation est de reconstruire l'ordre des marqueurs à partir de la matrice de criblage hybride (Slonim *et al.*,1997 [317]).

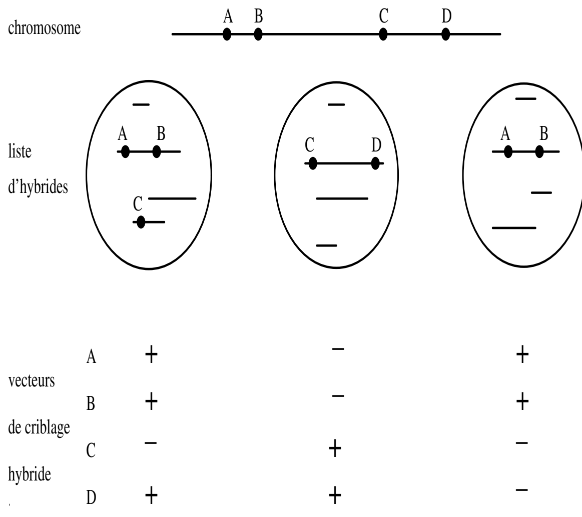


Figure 3.8 – Matrice de criblage hybride.

Si les cassures provenant des irradiations apparaissent aléatoirement de manière uniforme au travers du chromosome, alors les cassures entre des marqueurs situés à proximité l'un de l'autre (comme A et B) sont rares. Ceci implique qu'ils sont « co-conservés », c'est-à-dire que la cellule hybride contient soit les deux marqueurs, soit aucun des deux. Cette observation permet d'en déduire les marqueurs situés à proximité et d'utiliser des arguments similaires à la cartographie génétique pour l'ordonnancement des sondes (Boehnke *et al.*, 1991 [39], Lange *et al.*, 1995 [215]). Slonim *et al.*, 1997 [317] ont utilisé l'approche des modèles de Markov pour la cartographie RH et ont construit le squelette d'une carte RH en examinant les triplets de marqueurs. L'ordre le plus probable des triplets de marqueurs peut être estimé à partir de la matrice de criblage hybride. L'exemple de la figure 3.8 peut aboutir à un sous-ensemble de triplets ABC, ABD, ACD et BCD et le problème consiste à trouver une chaîne (ABCD) qui contient ces triplets comme sous-séquences. Ce problème est compliqué par la présence de triplets mal ordonnés et par l'orientation inconnue de ces derniers : l'ordre des marqueurs (A-B-C ou C-B-A) pour le triplet ABC n'est pas connu.

Chapitre 4

Séquençage

4.1 Introduction

À une époque où le séquençage du génome humain est presqu'achevé, peu de gens se souviennent que, avant le séquençage de l'ADN, les scientifiques séquençaient des protéines. Frederick Sanger a obtenu son premier prix Nobel pour avoir déterminé la séquence d'acides aminés de l'insuline, hormone hypoglycémiante utilisée pour traiter le diabète. À la fin des années 40, le séquençage de l'insuline semblait être un plus grand défi que ne l'est actuellement le séquençage de l'intégralité d'un génome bactérien. Les aspects informatiques du séquençage protéique à cette époque sont très similaires à ceux du séquençage moderne de l'ADN. La différence tient principalement à la longueur des fragments séquencés. À la fin des années 40, les biologistes ont appris à les biologistes ont appris à rompre la liaison entre l'acide aminé situé en N terminal de la protéine et le reste de la chaîne polypeptidique puis à déterminer sa nature. Cependant, la réitération de cette approche ne donnait de renseignements que sur quelques acides aminés car, après quatre ou cinq cycles, les résultats étaient difficiles à interpréter. Pour passer outre ce problème, Sanger a digéré l'insuline avec des protéases et a séquencé chaque fragment obtenu. Ensuite, il a utilisé ces fragments chevauchants pour reconstruire la séquence entière, exactement comme dans la méthode actuelle de séquençage de l'ADN, « casser - lire les fragments - assembler » :

Gly	*Ile*	*Val*	*Glu*				
	Ile	*Val*	*Glu*	*Gln*			
				Gln	*Cys*	*Cys*	*Ala*
Gly	*Ile*	*Val*	*Glu*	*Gln*	*Cys*	*Cys*	*Ala*

La dégradation d'Edman, qui détermine la nature de l'acide aminé terminal après clivage de la liaison peptidique entre ce dernier et le reste de la protéine, est devenue la méthode de séquençage protéique dominante de ces vingt

dernières années et, depuis la fin des années 60, des machines de séquençage protéique sont sur le marché.

La méthode de séquençage protéique de Sanger a influencé le travail sur le séquençage de l'ARN. Le premier ARN a été séquencé en 1965, avec la même approche « casser - lire les fragments - assembler ». Il a fallu sept ans à Holley et à ses collaborateurs de l'Université de Cornell pour déterminer la séquence de 77 nucléotides dans l'ARNt. Ensuite, durant de nombreuses années, le séquençage de l'ADN a été réalisé en transcrivant d'abord l'ADN en ARN, puis en séquençant l'ARN.

Les méthodes de séquençage de l'ADN ont été inventées de façon indépendante et simultanée à Cambridge, en Angleterre et à Cambridge, dans le Massachusetts. En 1974, le scientifique russe Andrey Mirzabekov, qui faisait un séjour au laboratoire de Walter Gilbert à l'Université de Harvard, a découvert des méthodes de clivage chimique de l'ADN permettant de couper ce dernier au niveau des A ou des G. Par la suite, Maxam et Gilbert ont étendu cette méthode et réussi à cliver l'ADN au niveau des C ou des T. Après avoir mesuré les longueurs des fragments obtenus dans quatre réactions séparées, ils ont pu séquencer l'ADN.

La méthode de Sanger tire profit de la façon dont les cellules font des copies de l'ADN. Les cellules copient l'ADN base après base grâce à une enzyme, l'ADN polymérase, qui ajoute les bases « complémentaires » les unes après les autres sur la chaîne nucléotidique en cours de synthèse ; Sanger a réalisé qu'il pouvait faire une échelle de fragments d'ADN de différentes tailles s'il « affamait » la réaction de l'une des quatre bases nécessaires pour faire de l'ADN. L'enzyme copierait l'ADN en stoppant de manière aléatoire la réaction de polymérisation au moment de l'incorporation de la base limitante, générant ainsi tous les fragments qui auraient dû se terminer par cette base. Ainsi, pour une séquence ACGTAAGCTA, le fait d'affamer en T produirait les fragments ACG et ACGTAAGC. En réalisant quatre expériences d'inanition pour A, T, G et C et en séparant les fragments d'ADN obtenus selon leur longueur, on peut lire l'ADN. Par la suite, Sanger a trouvé des analogues de chacune des bases qui, une fois insérés à leur place, empêchaient toute élongation du brin d'ADN. Ainsi, l'utilisation d'un analogue permettait de générer tous les fragments se terminant par la base correspondante. Ensuite, à partir de 1977, deux techniques indépendantes de séquençage de l'ADN ont été développées (Sanger *et al.*, 1977 [297] et Maxam et Gilbert, 1977 [233]), qui ont abouti au séquençage d'un virus de 5 386 nucléotides, ainsi qu'à un Prix Nobel en 1980. Depuis, la quantité de données pour le séquençage de l'ADN a augmenté de façon exponentielle et, en 1989, le projet du génome humain a été lancé. Le but est de déterminer l'ordre des trois milliards de nucléotides que contient le génome humain et d'obtenir ainsi des renseignements sur les gènes codés par ce génome (environ 30 000). Il est vraisemblable que les textes génétiques deviendront les principaux outils de recherche des biologistes au cours des prochaines décennies.

De la même façon que pour le séquençage protéique il y a 50 ans, les biologistes modernes ne sont capables que de définir la séquence de courts fragments

d'ADN (300 à 500 nucléotides) ; il faut donc définir une stratégie permettant d'obtenir la séquence d'un très grand fragment d'ADN, voire d'un génome entier. L'approche aléatoire (shotgun) conventionnelle s'appuie sur la fragmentation aléatoire d'un grand fragment d'ADN en petits, qui seront clonés individuellement dans un vecteur, afin de pouvoir les amplifier (en obtenir une quantité suffisante) et les séquencer. L'échantillon (grand fragment) est cassé mécaniquement (par un traitement aux ultrasons, par exemple), afin d'obtenir des coupures aléatoires. Les petits fragments d'une taille optimale sont sélectionnés et clonés dans un vecteur (on les appelle alors des *inserts*), le tout étant transféré dans un hôte bactérien, chaque bactérie ne contenant qu'un couple vecteur/insert. Après s'être reproduite, chaque bactérie forme une colonie bactérienne qui contient des millions de copies du vecteur, ainsi que l'insert qui lui est associé. Ainsi, le clonage et l'amplification dans l'hôte bactérien aboutissent à la sélection et à l'amplification d'un couple vecteur/insert et donc d'un insert donné qui est séquencé, typiquement selon la méthode de Sanger *et al.*, 1977 [297]. Comme seul un nombre limité de nucléotides de l'insert (reads) peut être déterminé à partir d'une réaction de séquençage, il est nécessaire d'assembler tous ces reads afin de reconstruire la séquence complète du grand fragment d'ADN de départ. Pour assembler ces fragments, les biologistes doivent résoudre un problème délicat, pas très différent de celui qui consiste à assembler le texte d'un livre à partir de nombreux morceaux de papiers.

4.2 Chevauchement, agencement et consensus

Après avoir séquencé de courts fragments d'ADN (des *reads*), on désire donc les assembler et reconstruire l'intégralité de la séquence d'ADN du clone (*problème d'assemblage de fragments*). Le problème de la plus courte super-chaîne est une abstraction extrêmement simplifiée qui ne capte pas le problème réel d'assemblage de fragments, car il suppose des données parfaites et peut réduire les répétitions d'ADN. Le génome humain contient de nombreuses répétitions ; par exemple, les séquences Alu de 300 bp sont présentes environ un million de fois dans le génome humain. Cependant, des variations de la séquence de ces répétitions ont été accumulées durant l'évolution. Le fait qu'elles ne soient pas tout à fait identiques nous donne une chance de réaliser l'assemblage, même en présence de ces répétitions. L'ADN est constitué de deux brins complémentaires, comme l'ont décrit Watson et Crick. Ainsi, celui des deux qui est réellement représenté dans un read dépend de la façon arbitraire dont l'insert est orienté dans le vecteur. L'appartenance à l'un ou l'autre des deux brins d'ADN constitue donc une complication supplémentaire dans le problème d'assemblage de fragments.

Les premiers algorithmes de séquençage suivaient la stratégie gloutonne et fusionnaient les chaînes (en commençant par celles qui avaient les chevauchements les plus longs), jusqu'à ce qu'il n'en reste qu'une. La plupart des algorithmes d'assemblage de fragments incluent les trois étapes suivantes (Peltola

et al., 1984 [262] et Kececioglu et Myers, 1995 [195]) :

- *Chevauchement.* Trouver les éventuels fragments chevauchants.
- *Agencement.* Trouver l'ordre des fragments.
- *Consensus.* Déduire de l'agencement la séquence d'ADN.

Le problème du chevauchement consiste à trouver la meilleure alliance entre le suffixe d'une séquence et le préfixe d'une autre (ou le préfixe de son complémentaire, voir ci-dessus). S'il n'y avait pas d'erreur de séquençage, on trouverait simplement le plus long suffixe d'une chaîne qui s'assortit exactement au préfixe d'une autre. Cependant, les erreurs de séquençage nous obligent à utiliser une variation de l'algorithme sous forme de programmation dynamique pour l'alignement de séquences. Comme les erreurs sont faibles (de 1 à 3%), la pratique courante est d'utiliser des méthodes de filtrage et de séparer les paires de fragments n'ayant pas de sous-chaîne commune suffisamment longue.

Construire l'agencement est l'étape la plus difficile dans l'assemblage de séquences d'ADN. On peut voir une séquence d'ADN d'un fragment comme une vaste empreinte de celui-ci et utiliser les idées informatiques de l'assemblage de cartes. De nombreux algorithmes d'assemblage de fragments choisissent une paire de fragments ayant le meilleur chevauchement à chaque étape. Le score de chevauchement est soit le score de similitude, soit un score probabiliste plus évolué comme dans le programme Phrap (Green, 1994 [136]). On vérifie la cohérence de la paire de fragments choisie et, si cette vérification est acceptée, les deux fragments sont fusionnés. Durant les dernières phases de l'algorithme, les collections de fragments — plutôt que les fragments individuels — sont fusionnées. Les différentes étapes de fusions de fragments génèrent ce que l'on appelle des *contigs.* La difficulté de l'étape d'agencement est de décider si deux fragments ayant un bon chevauchement se chevauchent vraiment (autrement dit, leurs différences sont causées par les erreurs de séquençage) ou représentent une répétition dans un génome (leurs différences proviennent de mutations).

La façon la plus simple de construire le consensus est de reporter le caractère le plus fréquent dans l'agencement de la sous-chaîne qui est (implicitement) construite une fois l'étape d'agencement finie. Des algorithmes plus sophistiqués alignent de façon optimale les sous-chaînes dans de petites fenêtres le long de l'agencement. Quant à Phrap (Green, 1994 [136]), il construit la séquence consensus comme une mosaïque des meilleurs segments (en termes de certains scores probabilistes) de l'agencement.

4.3 Shotgun avec séquençage des deux extrémités d'un même insert

Au début des années 80, la taille des fragments d'ADN utilisés dans une approche shotgun telle que décrite précédemment, n'était encore que de quelques centaines de milliers de bases. Ainsi, le projet du séquençage du génome humain a suivi initialement cette stratégie. Dans un premier temps, le génome est découpé en grands fragments d'ADN qui sont clonés dans un vecteur adéquat. Ensuite, on choisit un ensemble minimal de clones formant un pavage qui recouvre le génome en réalisant une cartographie physique. Finalement, chacun des clones est séquencé en utilisant une approche shotgun.

Au milieu des années 90, le séquençage du génome de la bactérie *H. Infuenza* (1800 Kb) a été obtenu par une approche shotgun directement réalisée sur le génome entier. Inspirés par cette percée, Weber et Myers, 1997 [367] ont proposé d'utiliser l'approche shotgun pour séquencer le génome humain entier. Un an plus tard, une société du nom de Celera Genomics a été créée dans le but d'effectuer le séquençage shotgun du génome humain pour l'année 2001.

Cependant, l'assemblage de fragments devient très difficile dans des projets de séquençage à grande échelle, comme le séquençage du génome de la drosophile. Dans ce cas, les algorithmes d'assemblage de fragments standards ont tendance à réduire les répétitions qui se trouvent dans différentes parties du génome. Augmenter la longueur du read résoudrait le problème, mais la technologie de séquençage n'a pas encore amélioré de façon significative cette longueur. Pour surmonter ce problème, les biologistes ont suggéré un doublement virtuel de la longueur de lecture en obtenant une paire de reads correspondants à la séquence *des deux* extrémités d'un même insert. Ceci produit une paire de reads (appelés des *conjugués*) d'orientations opposées, situés à une distance approximative connue l'un de l'autre. En utilisant des inserts de tailles variables (jusqu'à plusieurs dizaines, voire centaines, de kb), les séquences des extrémités d'un même fragment, si elles sont retrouvées dans des contigs différents, permettent de relier physiquement ces contigs et d'avoir une notion de la taille de la brèche les séparant. On obtient ainsi des *scaffolds.*

Les répétitions représentent un défi majeur pour le séquençage shotgun du génome entier. Elles apparaissent à plusieurs échelles. Par exemple, dans le locus du récepteur T humain, il y a cinq répétitions situées à proximité du gène trypsinogène, long de 4 Kb, qui varie de 3 à 5% entre les copies. Ces répétitions sont difficiles à assembler car les reads ayant des portions uniques à l'extérieur de la répétition ne peuvent les franchir. Le génome humain contient un nombre de répétitions *Alu* (300 bp) estimé à un million et 200 000 répétitions *LINE* (1000 bp), sans parler des 25% de gènes humains qui, selon les estimations, sont présents dans au moins deux copies.

L'avantage informatique du séquençage des deux extrémités d'un insert est qu'il est peu probable que les deux reads de l'insert se trouvent dans une répétition d'ADN de grande échelle (Roach *et al.*, 1995 [286]). Dans une portion

d'ADN unique, le read détermine donc dans quelle copie d'une répétition se trouve son conjugué.

Cette approche peut être rendue plus puissante par l'utilisation des cartes STS pour l'assemblage de fragments. Le STS est un fragment d'ADN unique de 300 bp et les cartes STS valables ordonnent des dizaines de milliers de STS le long des chromosomes humains (Hudson *et al.*, 1995 [173]). Comme la distance approximative entre des STS consécutifs est connue, les positions des STS peuvent servir de points de contrôle dans l'assemblage de fragments (Weber et Myers, 1997 [367] et Myers, 1999 [247]).

Les ambitieux projets de séquençage génomique utilisant une telle approche lancent le défi consistant à « finir » le séquençage dans les secteurs qui n'ont pas été couverts après la fin de l'étape shotgun. Toute quantité raisonnable de séquençage shotgun va laisser des secteurs séquencés de manière insuffisante. Ceux-ci incluront à la fois des secteurs séquencés avec une faible couverture et des brèches de longueur inconnue. L'achèvement est obtenu grâce à des expériences de *marche* qui utilisent les amorces provenant des contigs déjà séquencés pour étendre une région séquencée d'une longueur d'un read. L'optimisation du séquençage de l'ADN requiert un échange entre la quantité de séquençage shotgun et les expériences de marche. Un autre problème est de savoir où marcher afin de rencontrer des critères de recouvrement minimal.

4.4 Quelques autres problèmes et approches

4.4.1 Problème de la plus courte super-chaîne

Blum *et al.*, 1994 [37] ont imaginé un algorithme qui trouve une super-chaîne qui ne fait pas trois fois la longueur optimale. Par la suite, Breslauer *et al.*, 1997 [48] ont décrit un algorithme avec un rapport d'approximation de $2,596$. La question concernant le rapport d'approximation d'un algorithme glouton simple qui fusionne une paire de chaînes avec un chevauchement maximal reste ouverte. Personne n'a produit d'exemple montrant que cet algorithme produit une super-chaîne plus de deux fois plus longue que l'optimale. La conjecture est donc que l'algorithme glouton est une 2-approximation.

4.4.2 Phase d'achèvement du séquençage d'ADN

Le minimum requis pour la production de séquences exactes d'ADN est d'avoir au moins trois clones couvrant chaque position d'ADN et d'utiliser la règle majoritaire dans la constrution du consensus. Cependant, tout projet de séquençage génomique est susceptible d'aboutir à des fragments d'ADN ayant un faible recouvrement clonal. Après avoir établi les localisations de ces régions, on a besoin d'une phase supplémentaire d'achèvement qui est habituellement réalisée par la marche génomique. L'ensemble des reads d'ADN séquencés définit un recouvrement de position qui correspond au nombre de reads recouvrant

la position x dans l'ADN. Étant donnée une condition de recouvrement redondant au k-uple, le but de la phase d'achèvement est de trouver un ensemble minimal de reads séquencés qui augmente le recouvrement de k pour chaque position (Czabarka *et al.*, 2000 [78]).

Chapitre 5

Puces à ADN

5.1 Introduction

Lorsque le projet du génome humain a commencé, le séquençage de l'ADN tenait de la routine, mais il s'agissait d'une procédure qui prenait du temps et qui était difficile à automatiser. En 1988, quatre groupes de biologistes ont suggéré de façon indépendante et simultanée une technique de séquençage totalement nouvelle appelée le *séquençage par hybridation* (SBH pour *Sequencing by Hybridization*). Le SBH implique la construction d'une *puce à ADN* miniature contenant des milliers de courts fragments d'ADN attachés sur sa surface : les sondes. Chacun de ces courts fragments révèle de l'information sur un fragment d'ADN inconnu ; une fois combinés, ils sont censés séquencer les fragments d'ADN. En 1988, presque personne ne croyait que l'idée fonctionnerait ; les problèmes biochimiques (la synthèse de milliers de courts fragments d'ADN sur la puce) et les problèmes combinatoires (la reconstruction de la séquence à partir des résultats de la puce) semblaient trop compliqués. Peu de temps après la publication du premier article décrivant les puces à ADN, le magazine *Science* écrivait que, étant donnée la quantité de travail nécessaire à la synthèse d'une puce à ADN, *« cela ne ferait que substituer une tâche fastidieuse à une autre »*. Il ne s'agissait pas là d'un bon pronostic : Fodor *et al.*, 1991 [110] ont réalisé une percée majeure dans la technologie des puces à ADN. Leur approche de la fabrication de puces est fondée sur la synthèse d'un polymère dirigée par la lumière, qui présente de nombreuses ressemblances avec la fabrication de puces pour ordinateur. Grâce à cette technique, la construction d'une puce avec les 4^l sondes de longueur l ne requiert que $4l$ réactions séparées. Avec cette méthode, Affymetrix, une compagnie de biotechnologie située en Californie, a fabriqué la première puce à ADN de 64 kb en 1994. Peu de temps après, la construction de puces de 1 Mb devenait une routine et l'idée des puces à ADN cessait d'être un jeu intellectuel pour devenir l'une des biotechnologies les plus prometteuses, permettant de révolutionner les diagnostics médicaux et la génomique fonctionnelle.

Toute sonde p dans une puce à ADN nous informe sur un fragment d'ADN cible (inconnu) en répondant à la question de savoir si p s'hybride à ce fragment. Étant donné un fragment d'ADN inconnu, une puce fournit de l'information sur toutes les chaînes de longueur l contenues dans ce fragment (décomposition en l-uplets du fragment), mais elle ne donne pas d'information sur les positions de ces chaînes. On utilise donc des algorithmes combinatoires pour reconstruire la séquence du fragment à partir de sa décomposition en l-uplets.

Le problème SBH peut être vu comme le problème du chemin hamiltonien ; ce problème consiste à trouver dans un graphe un chemin qui passe exactement une fois par chaque sommet. Les sommets du graphe correspondent aux l-uplets et les arêtes aux paires de l-uplets qui se chevauchent. Cependant, cette réduction ne mène pas à un algorithme SBH efficace, car on ne connaît pas d'algorithme efficace pour le problème du chemin hamiltonien. En réalité, le problème SBH a été résolu il y a longtemps — et même des siècles avant l'existence d'études de biologie moléculaire — par... Leonhard Euler, le grand mathématicien du XVIIIe siècle. Évidemment, il ne savait pas qu'il résolvait le problème SBH ; il essayait juste de résoudre le problème des « sept ponts de Königsberg ». La ville de Königsberg contenait une île reliée à la terre par sept ponts (voir figure 5.1) ; Euler cherchait un chemin passant par chaque pont exactement une fois. La solution à ce problème annonçait la naissance de la théorie des graphes et, deux siècles plus tard, aboutissait à la solution de nombreux problèmes combinatoires ; le SBH en fait partie.

Bien que les puces à ADN aient été initialement proposées comme une alternative au séquençage conventionnel sur gel de l'ADN, le séquençage *de novo* avec des puces à ADN demeure un problème difficile. Les premiers obstacles dans les applications des puces à ADN sont les inexactitudes dans l'interprétation des données d'hybridation : la distinction entre de parfaits appariements et de mauvais appariements très stables. Ceci constitue un problème particulièrement difficile pour les sondes courtes (8 à 10 nucléotides) utilisées dans le séquençage *de novo*.

Par suite, les puces à ADN ont trouvé davantage d'applications dans le *re-séquençage* et la *détection de mutations* (qui peuvent être réalisés avec des sondes plus longues) que dans le séquençage *de novo*. Dans ce cas, le problème est de trouver les différences entre le gène de type sauvage (connu) et le gène muté (utile). On peut créer des sondes relativement longues (20 nucléotides) pour détecter de façon certaine les mutations et pour contourner le problème encore non résolu consistant à distinguer les parfaits appariements des mauvais très stables. Ces sondes sont habituellement des variations des sondes qui s'hybrident au fragment d'ADN connu. Par exemple, chaque 20-uplet dans l'ADN peut correspondre à quatre sondes : le type sauvage et trois *mutations médianes* dont une position centrale a été remplacée par l'un des autres nucléotides. Lipshutz *et al.*, 1995 [226] ont décrit de telles *puces de pavage* pour détecter les mutations dans le virus VIH. Bien qu'aucun signal du seuil d'hybridation ne puisse distinguer des appariements parfaits et imparfaits, la distinction entre ces signaux est obtenue en comparant les intensités d'hybridation d'une sonde

avec les intensités d'hybridation de ses mutations médianes.

Les puces de pavage peuvent être utilisées pour explorer la diversité génétique de populations entières. L'analyse des mutations dans l'ADN mitochondrial humain a fortement influencé l'étude de l'évolution de l'être humain et des maladies génétiques. Ces études impliquent le re-séquençage de l'ADN mitochondrial humain chez de nombreux individus pour trouver les mutations. À cause du coût du séquençage conventionnel, la plupart des études se sont limitées à de petites régions hypervariables totalisant environ 600 paires de bases. Chee *et al.*, 1996 [65] ont décrit une puce de pavage pour le génome mitochondrial humain complet (16 569 paires de bases) et ont pu détecter trois mutations à l'origine de la maladie dans un échantillon d'ADNmt d'un patient atteint de neuropathie optique héréditaire de Leber. Les puces d'ADNmt préfabriquées nous permettent de re-séquencer l'ADN chez de nombreux individus et fournissent une technologie efficace et rapide pour les études sur l'évolution moléculaire (Hacia *et al.*, 1999 [148]). D'autres applications des puces à ADN incluent la génomique fonctionnelle (surveillance continue de l'expression génétique) et la cartographie génétique (Golub *et al.*, 1999 [131] et Wang *et al.*, 1998 [349]).

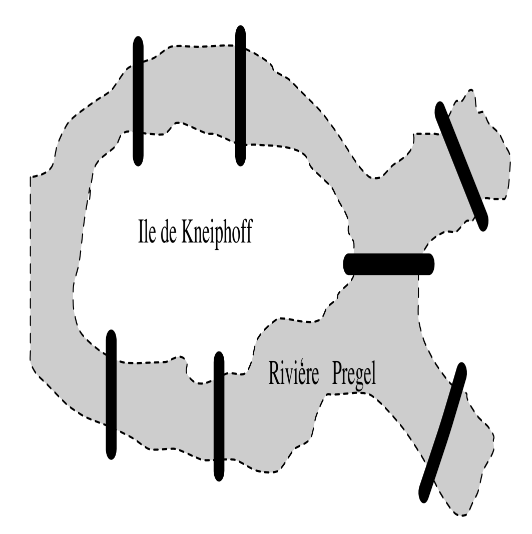


Figure 5.1 – Ponts de Königsberg.

5.2 Séquençage par hybridation

Les *puces à ADN* ont été suggérées de façon simultanée et indépendante par Bains et Smith, 1988 [23], Drmanac *et al.*, 1989 [91], Lysov *et al.*, 1988 [228] et Southern, 1988 [325]. Les inventeurs des puces à ADN ont proposé de les utiliser pour le séquençage de l'ADN et cette technologie s'appelait, à l'origine, le

séquençage de l'ADN par hybridation (SBH). Le SBH repose sur l'hybridation d'un fragment (inconnu) d'ADN avec de grandes rangées de courtes sondes. Étant donné un court fragment synthétique d'ADN (8 à 30 nucléotides), que l'on appelle une *sonde*, et un fragment d'ADN cible à simple brin, la cible va s'attacher (*s'hybrider*) à la sonde si elle possède une sous-chaîne qui est le *complément* de Watson-Crick de la sonde (A est le complémentaire de T et G est le complémentaire de C). Par exemple, une sonde $ACCGTGGA$ va s'hybrider à une cible CCC$TGGCACCT$A, car elle est complémentaire à la sous-chaîne $TGGCACCT$ de la cible. De cette façon, les sondes peuvent être utilisées pour tester la cible d'ADN inconnue et déterminer sa décomposition en l-uplets. La puce à ADN la plus simple, $C(l)$, contient toutes les sondes de longueur l et fonctionne comme suit (voir figure 5.2) :

- Attacher toutes les sondes possibles de longueur l (l=8 dans les premiers articles sur le SBH) à la surface, chaque sonde se trouvant à un endroit distinct connu. Cet ensemble de sondes est appelé la *puce à ADN*.

- Appliquer à la puce une solution contenant un fragment d'ADN lié à un marqueur fluorescent.

- Le fragment d'ADN s'hybride aux sondes qui sont complémentaires des sous-chaînes de longueur l de ce fragment.

- Détecter les sondes qui s'hybrident au fragment d'ADN (à l'aide d'un détecteur spectroscopique) et obtenir la décomposition en l-uplets du fragment d'ADN.

- Appliquer un algorithme combinatoire pour reconstruire la séquence du fragment d'ADN à partir de la décomposition en l-uplets.

La puce à ADN présentant tous les 8-uplets, $C(8)$, requiert la synthèse de $4^8 = 65\,536$ sondes. Cela constituait vraiment une tâche difficilement réalisable en 1988, lorsque les puces à ADN ont été proposées la toute première fois.

5.3 SBH et problème de la plus courte super-chaîne

Le SBH fournit de l'information sur les l-uplets présents dans l'ADN, mais il ne renseigne pas sur leurs positions. Supposons que l'on connaisse toutes les sous-chaînes de longueur l d'une chaîne inconnue (décomposition en l-uplets ou *spectre* du fragment d'ADN). Comment peut-on reconstruire le fragment d'ADN cible à partir de ces données ?

Le SBH peut être considéré comme un cas particulier du *problème de la plus courte super-chaîne*. Pour un ensemble donné de chaînes $s_1, \ldots, s_m$, une

PUCE A ADN C(4)

	AA	AT	AG	AC	TA	TT	TG	TC	GA	GT	GG	GC	CA	CT	CG	CC
AA																
AT			ATAG													
AG																
AC												ACGC				
TA											TAGG					
TT																
TG																
TC																
GA																
GT																
GG													GGCA			
GC	GCAA															
CA	CAAA															
CT																
CG																
CC																

l'ADN cible TATCCGTTT (complémentaire de ATAGGCAAA)
s'hybride à la puce de tous les 4-meres :

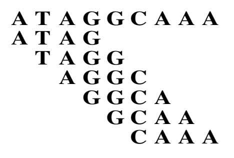
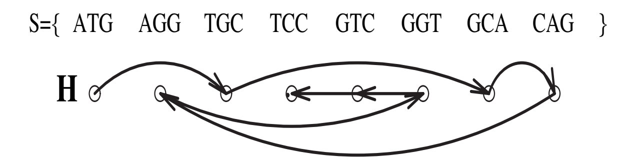


Figure 5.2 – Hybridation de TATCCGTTT avec la puce à ADN $C(4)$.

CONSTRUCTION DE SEQUENCE (approche du chemin hamiltonien)



Sommets : l-uplets du spectre S. Arêtes : l-uplets qui se chevauchent.

Le chemin qui passe par TOUS LES SOMMETS correspond à la reconstruction de la séquence :

ATGCAGGTCC

Figure 5.3 – Le SBH et le problème du chemin hamiltonien.

super-chaîne est une chaîne qui contient chaque s_i comme sous-chaîne. Étant donné un ensemble de chaînes, le problème consistant à trouver la chaîne la plus courte est NP-complet (Gallant *et al.*, 1980 [117]).

On définit $chevauchement(s_i, s_j)$ comme étant la longueur du préfixe maximal de s_j qui s'apparie à un suffixe de s_i. Le problème de la plus courte super-chaîne peut être vu comme le problème du voyageur de commerce dans un graphe complet orienté avec m sommets qui correspondent aux chaînes s_i et des arcs (qui sont des arêtes orientées) de longueur $-chevauchement(s_i, s_j)$. Le SBH correspond au cas particulier dans lequel toutes les sous-chaînes $s_1, \ldots, s_m$ ont la même longueur. Les l-uplets p et q *se chevauchent* si les $l-1$ dernières lettres de p coïncident avec les $l-1$ premières lettres de q, c'est-à-dire $chevauchement(p, q) = l - 1$. Étant donné le spectre S d'un fragment d'ADN, on construit le graphe orienté H d'ensemble de sommets S et d'ensemble d'arcs $E = \{(p, q) : p \text{ et } q \text{ se chevauchent}\}$. Le graphe H est formé des arcs les plus légers (de longueur $-(l-1)$) du graphe complet orienté défini précédemment. Il existe une correspondance bijective entre les chemins qui passent au moins une fois par chaque sommet de H et les fragments d'ADN de spectre S. Le spectre présenté en figure 5.3 donne un *graphe-chemin* H. Dans ce cas, la reconstruction de la séquence ATGCAGGTCC correspond à l'unique chemin

$$ATG \to TGC \to GCA \to CAG \to AGG \to GGT \to GTC \to TCC,$$

qui passe par tous les sommets de H. Un chemin qui visite chaque sommet d'un graphe exactement une fois est appelé un chemin *hamiltonien*. Le spectre montré dans la figure 5.4 donne un graphe plus compliqué avec deux chemins hamiltoniens qui correspondent à deux reconstructions possibles.

Pour des fragments d'ADN plus grands, les graphes de chevauchement deviennent plutôt compliqués et difficiles à analyser. Le problème du chemin hamiltonien est NP-complet et on ne connaît aucun algorithme efficace per-

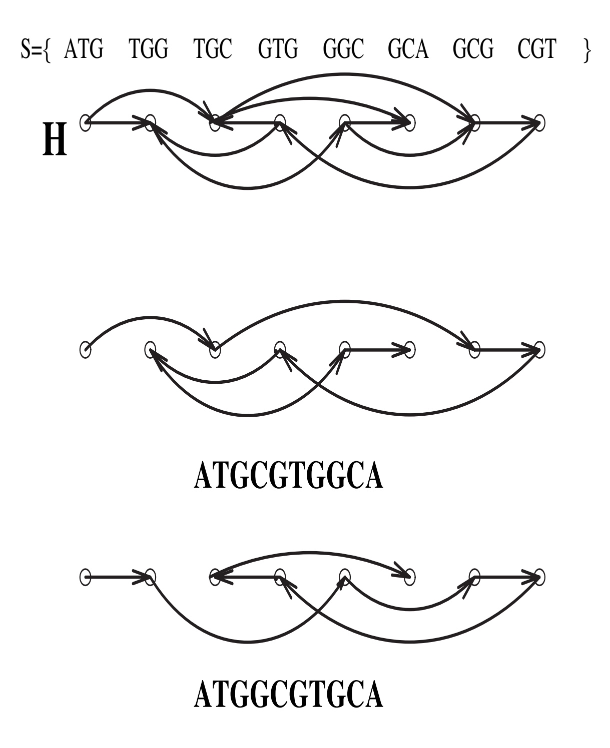


Figure 5.4 – Le spectre S donne deux reconstructions possibles qui correspondent à des chemins hamiltoniens distincts.

mettant de le résoudre. Par suite, l'approche décrite n'est pas pratique pour de longs fragments d'ADN.

5.4 SBH et problème du chemin eulérien

Comme on vient de le voir, réduire le problème SBH à celui du chemin hamiltonien n'aboutit pas à des algorithmes efficaces. Heureusement, si l'on se ramène au problème du *chemin eulérien*, on obtient un algorithme simple de complexité linéaire pour la reconstruction de séquences. L'idée de cette approche est de construire un graphe dont les arcs (plutôt que les sommets, comme dans la construction précédente) correspondent aux l-uplets et de trouver un chemin qui passe par chaque arc exactement une fois. Dans cette approche, on construit un graphe G sur l'ensemble de tous les $(l-1)$-uplets (Pevzner, 1989 [264]). Un $(l-1)$-uplet v est relié par un arc à un $(l-1)$-uplet w si le spectre contient un l-uplet pour lequel les $l-1$ premiers nucléotides coïncident avec v et les $l-1$ derniers avec w (voir figure 5.5). Chaque élément du spectre

correspond à un arc dans G, et non à un *sommet* comme dans H (comparer les figures 5.3 et 5.5). Par conséquent, trouver un fragment d'ADN contenant toutes les sondes du spectre correspond à trouver un chemin passant par tous les *arcs* de G, autrement dit un chemin *eulérien.* Trouver des chemins eulériens est un problème bien connu et simple.

Un graphe orienté G est dit *eulérien* s'il contient un cycle qui traverse chaque arc de G exactement une fois. Un sommet v d'un graphe est dit *équilibré* si le nombre d'arcs arrivant en v est égal au nombre d'arcs qui en sortent. On note $\deg_G^-(v)$ le nombre d'arcs qui arrivent en v ; on l'appelle le *degré entrant* (ou *demi-degré intérieur*) de v. De la même façon, le nombre $\deg_G^+(v)$ est le *degré sortant* (ou *demi-degré extérieur*) de v. Dans le cas d'un sommet équilibré, on a donc : $\deg_G^+(v) = \deg_G^-(v)$. Le théorème suivant donne une caractérisation des graphes eulériens :

Théorème 5.1 *Un graphe connexe est eulérien si et seulement si chacun de ses sommets est équilibré.*

Le problème SBH équivaut à trouver un *chemin eulérien* dans un graphe. Un sommet v d'un graphe est dit *semi-équilibré* s'il vérifie l'égalité suivante : $|\deg_G^+(v) - \deg_G^-(v)| = 1$. Le problème du chemin eulérien peut se ramener au problème du cycle eulérien en ajoutant un arc entre deux sommets semi-équilibrés.

Théorème 5.2 *Un graphe connexe possède un chemin eulérien si et seulement s'il contient au plus deux sommets semi-équilibrés et si tous les autres sont équilibrés.*

Pour construire un cycle eulérien, on part d'un arc quelconque de G et on forme un chemin construit aléatoirement en étendant le chemin existant avec de nouveaux arcs arbitrairement choisis. Cette procédure se termine lorsque tous les arcs incidents à un sommet de G sont utilisés dans le chemin. Comme chaque sommet de G est équilibré, tout chemin de ce type qui commence en v se termine en v. Avec un peu de chance, il sera eulérien, mais ce n'est pas nécessairement le cas. Si le chemin n'est pas eulérien, il doit contenir un sommet w incident à un certain nombre d'arcs non empruntés. Notons que tous les sommets situés dans le graphe des arcs non traversés sont équilibrés et que, par conséquent, il existe un chemin aléatoire qui commence et finit en w et qui ne contient que des arcs non traversés. On peut désormais agrandir le chemin aléatoire comme suit : on insère un chemin aléatoire d'arcs non traversés commençant en w à l'endroit où le chemin aléatoire qui débute en v atteint w. En répétant ceci, on peut obtenir un cycle eulérien. Cet algorithme peut être implémenté en temps linéaire (Fleischner, 1990 [108]).

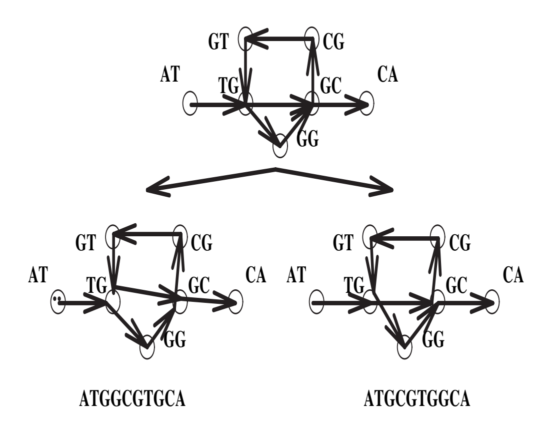
Le nombre de reconstructions de séquences différentes dans le SBH est borné par le nombre de cycles eulériens (chemins). La formule qui donne le nombre de cycles eulériens dans un graphe orienté est connu sous le nom de théorème *BEST* (Fleischner, 1990 [108]). Soit G un graphe eulérien orienté de matrice

RECONSTRUCTION DE SEQUENCE (approche du chemin eulérien)

S={ATG, TGG, TGC, GTG, GGC, GCA, GCG , CGT}

Les sommets correspondent aux (l-1)-uplets.

Les arêtes correspondent aux l-uplets du spectre.



Les chemins qui passent par TOUTES LES ARETES correspondent aux reconstructions de séquences

Figure 5.5 – Le SBH et le problème du chemin eulérien.

d'adjacence $A = (a_{ij})$, où a_{ij} vaut 1 s'il existe un arc allant du sommet i au sommet j dans G et 0 sinon (voir figure 5.6). On définit M comme étant la matrice obtenue en remplaçant le i-ième terme de la diagonale de $-A$ par $\deg_G^-(i)$ pour tout i. Le i-ième cofacteur d'une matrice M est le déterminant dét (M_i) de la matrice M_i, obtenue à partir de M en supprimant sa i-ième ligne et sa i-ième colonne. Tous les cofacteurs de la matrice M ont la même valeur, notée $c(G)$.

Théorème 5.3 *Le nombre de cycles eulériens d'un graphe eulérien $G(V, E)$ est*

$$c(G) \times \prod_{v \in V} (\deg(v) - 1)!$$

Il existe une façon simple de vérifier si, pour un spectre donné, il existe une unique reconstruction de séquence. On décompose un graphe eulérien G en cycles *simples* $C_1, \ldots, C_t$, c'est-à-dire des cycles qui ne se recoupent pas. Chaque arc de G est utilisé dans exactement un cycle (les sommets de G peuvent

être utilisés dans plusieurs cycles). Pour ces cycles, on définit le graphe d'intersection G_I à t sommets $C_1, \ldots, C_t$, où C_i et C_j sont reliés par k arcs si et seulement s'ils ont k sommets communs.

Théorème 5.4 *Le graphe G possède un seul cycle eulérien si et seulement si le graphe d'intersection G_I est un arbre.*

G:

$$A=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad M=\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 2 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Figure 5.6 – Chaque cofacteur de la matrice M vaut 2. Le nombre de cycles eulériens dans le graphe G est $2 \times 0! \times 1! \times 1! \times 0! = 2$.

L'approche reposant sur le chemin eulérien fonctionne dans le cas d'expériences SBH sans erreur. Cependant, même dans ce cas, il peut exister des chemins eulériens multiples, aboutissant à des reconstructions multiples. Pour de réelles expériences avec des puces à ADN, les erreurs dans les spectres rendent la reconstruction encore plus difficile. En outre, les répétitions de longueur l compliquent l'analyse, car il est difficile de déterminer la multiplicité des l-uplets à partir des intensités d'hybridation.

La figure 5.7 présente le même spectre que dans la figure 5.5, avec deux trinucléotides manquants (faux négatif) et deux reconstructions possibles (seule l'une d'elles est correcte). La situation devient encore plus compliquée dans le cas de l'hybridation non spécifique, lorsque le spectre contient des l-uplets absents dans le fragment d'ADN cible (*faux positif*). Plusieurs approches biochimiques permettant l'élimination de l'hybridation non spécifique dans les expériences SBH tentent d'établir une meilleure distinction entre des appariements parfaits et imparfaits. Malgré tout, les données d'hybridation des puces à ADN restent encore bien trop ambiguës au goût des informaticiens et des biologistes.

5.5 Probabilité d'une reconstruction de séquence unique

Quelle est la probabilité qu'un fragment d'ADN de longueur n puisse être reconstruit de façon unique par une puce à ADN $C(l)$? Ou, en d'autres termes, combien doit valoir l pour reconstruire de façon unique une séquence aléatoire de longueur n à partir de son spectre de l-uplets ? Dans un souci de simplicité, supposons que les lettres du fragment d'ADN soient indépendantes et distri-

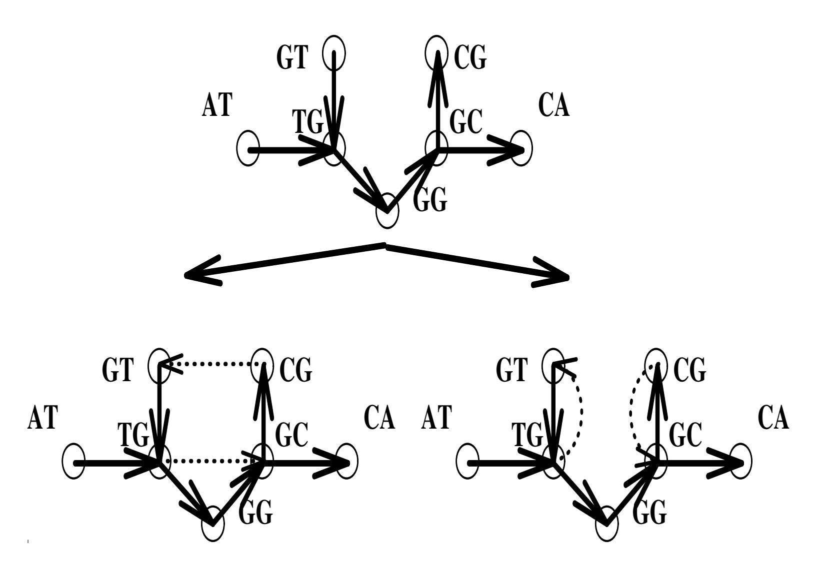


Figure 5.7 – Reconstruction de séquence dans le cas de l-uplets manquants (faux négatif).

buées de façon identique avec la probabilité $p = \frac{1}{4}$ pour chacun des A, T, G et C.

Une heuristique grossière est fondée sur l'observation selon laquelle les répétitions de l-uplets amènent souvent des reconstructions non uniques. Il y a environ $\binom{n}{2}p^l$ répétitions potentielles de longueur l correspondant aux paires de positions dans le fragment d'ADN de longueur n. En résolvant $\binom{n}{2}p^l = 1$, on obtient une estimation de $l = \log_{\frac{1}{p}}\binom{n}{2}$ pour la longueur minimale de la sonde nécessaire à une reconstruction certaine d'une séquence à n lettres à partir de son spectre de l-uplets. La longueur maximale du fragment d'ADN pouvant être reconstruit avec une puce à ADN $C(l)$ peut être estimée à $\sqrt{2 \times 4^l}$ approximativement.

Une analyse plus prudente révèle que les répétitions de l-uplets ne sont pas la cause majeure de reconstructions non uniques (Dyer *et al.*, 1994 [94]). La cause la plus probable est une paire intercalée de $(l-1)$-uplets répétés (dans la figure 5.9, les répétitions de AGTC et TTGG sont intercalées). Par conséquent, on devrait considérer les répétitions de longueur $l - 1$. On observe

aussi que les répétitions peuvent former des groupes ; il faudrait considérer également quelque chose comme les « répétitions maximales » de longueur $l-1$. Le nombre attendu de telles répétitions est d'environ $\lambda \approx \binom{n}{2}(1-p)p^{l-1}$. Une approximation de Poisson donne $P\{k \text{ répétitions}\} \approx e^{-\lambda}\frac{\lambda^k}{k!}$. Arratia *et al.*, 1996 [12] ont montré que, lorsqu'il y a k répétitions, la probabilité de n'avoir aucune paire intercalée est d'environ $\frac{k!2^k C_k}{(2k)!}$, où $C_k = \frac{1}{k+1}C_{2k}^k$ est le k-ième nombre de Catalan. En faisant la moyenne sur k, on obtient que la probabilité d'une reconstruction unique pour une séquence à n lettres à partir de son spectre de l-uplets est approximativement

$$\sum_{k\geq 0} e^{-\lambda}\frac{k!2^k C_k}{(2k)!}\frac{\lambda^k}{k!} = e^{-\lambda}\sum_{k\geq 0}\frac{(2\lambda)^k}{k!(k+1)!}.$$

Arratia *et al.*, 1996 [12] ont transformé ces arguments heuristiques en estimations précises pour la puissance de résolution des puces à ADN.

5.6 Réarrangements de chaînes

La figure 5.8 présente deux séquences d'ADN avec le même spectre SBH. Le graphe G correspondant au spectre de la figure 5.8 contient un sommet fourche TG. On ne sait pas quel triplet (TGC ou TGG) suit ATG dans la séquence originale et on ne peut faire la distinction entre la reconstruction correcte et la reconstruction incorrecte. Une expérience biochimique supplémentaire (l'hybridation d'un fragment d'ADN cible avec ATGC, par exemple) permettrait de trouver la reconstruction correcte (la séquence située en haut de la figure 5.8 contient ATGC, ce qui n'est pas le cas de la séquence du bas).

Des expériences biochimiques supplémentaires permettant de résoudre les fourches dans le processus de reconstruction ont d'abord été proposées par Southern, 1988 [325] (à l'aide d'une sonde plus longue pour chaque sommet fourche) et Khrapko *et al.*, 1989 [197] (*continuous stacking hybridization*). La technique proposée par Khrapko et ses collègues suppose une hybridation supplémentaire de courtes sondes qui étend continuellement les doubles formés par le fragment d'ADN cible et les sondes de la puce. Dans cette approche, l'hybridation supplémentaire avec un m-uplet sur la puce $C(l)$ fournit de l'information sur certains $(l+m)$-uplets contenus dans la séquence d'ADN cible. Les simulations informatiques suggèrent que cette technique permet, avec seulement trois expériences supplémentaires, une reconstruction non ambiguë d'un fragment de 1000 bp dans 97% des cas.

Pour analyser les expériences biochimiques supplémentaires, on a besoin d'une *caractérisation* de toutes les séquences d'ADN correspondant au spectre donné. Dans les toutes premières études sur les puces à ADN, les biologistes ont décrit des *réarrangements de chaînes* qui ne changent pas le spectre de ces chaînes (Drmanac et., 1989 [91]). Cependant, le problème consistant à caractériser *tous* ces réarrangements n'a toujours pas été résolu. Ukkonen, 1992 [340] a

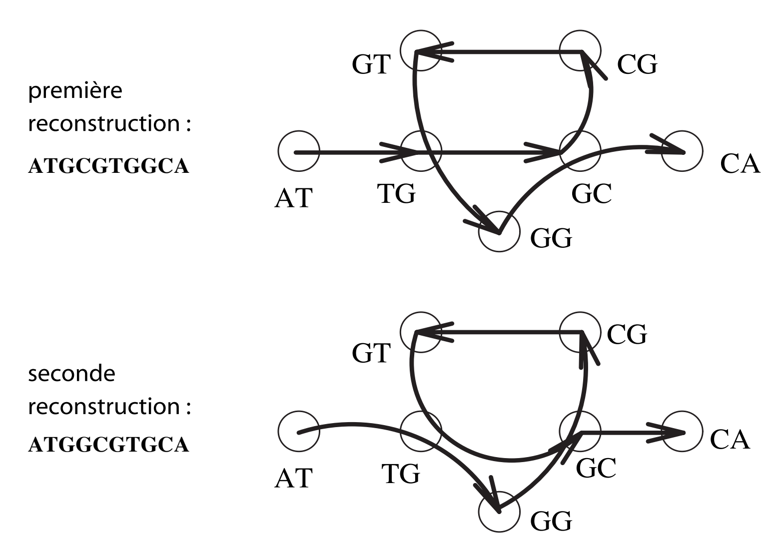


Figure 5.8 – Des expériences biochimiques supplémentaires lèvent les ambiguïtés dans la reconstruction de séquence.

conjecturé que, pour tout couple de chaînes ayant le même spectre de l-uplets, on peut passer de l'une à l'autre par des transformations simples appelées transpositions et rotations (voir figure 5.9). Par la suite, l'expression « une chaîne écrite sous forme de l-uplets » signifiera que la chaîne est écrite comme une suite de ses l-uplets. Par exemple, la notation en 2-uplets de la chaîne ATGGGC est AT TG GG GG GC. Ukkonen, 1992 [340] a conjecturé que, pour tout couple de chaînes ayant la même décomposition en l-uplets, on peut transformer l'une en l'autre par des opérations simples appelées transpositions et rotations.

Si une chaîne

$$s = \ldots x \underline{\ldots} z \ldots x \underbrace{\ldots} z \ldots$$

(écrite sous forme de $(l-1)$-uplets) contient des paires intercalées de $(l-1)$-uplets x et z, alors la chaîne $\ldots x \underbrace{\ldots} z \ldots x \underline{\ldots} z \ldots$, où $\underline{\ldots}$ et $\underbrace{\ldots}$ ont échangé leurs positions, est appelée une *transposition* de s. Si l'on a $s = \ldots x \underline{\ldots} x \underbrace{\ldots} x \ldots$, où x est un $(l-1)$-uplet, $\ldots x \underbrace{\ldots} x \underline{\ldots} x \ldots$ est également

Transformations de séquences équivalentes

gta AGTC cgggc TTGG taga AGTC aaattta TTGG at

gta AGTC aaattta TTGG taga AGTC cgggc TTGG at

transposition

AGTC gta TTGG taga AGTC

TTGG taga AGTC gta TTGG

rotation

Figure 5.9 – Transformations d'Ukkonen ($l = 5$).

appelée une transposition de s. Si une chaîne

$$s = x\underline{\ldots}z\underbrace{\ldots}x$$

(écrite sous forme de $(l-1)$-uplets) commence et se termine par le même $(l-1)$-uplet x, alors la chaîne $z\underbrace{\ldots}x\underline{\ldots}z$ est appelée une *rotation* de s.

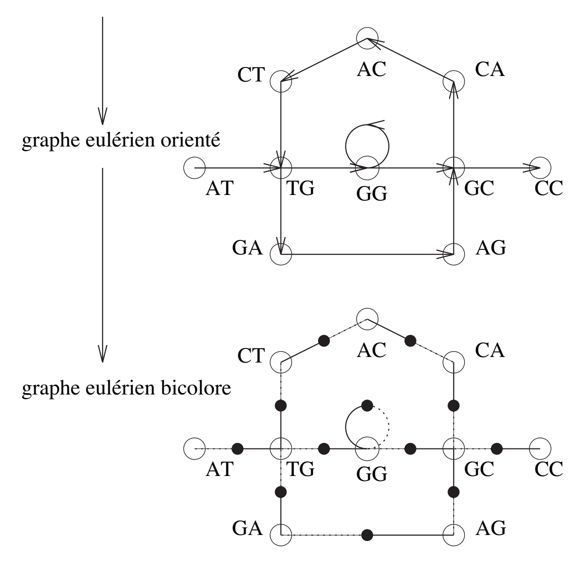
Il est clair que les transpositions et les rotations ne changent pas la décomposition en l-uplets.

Théorème 5.5 *(Pevzner, 1995 [267]) Pour tout couple de chaînes ayant la même décomposition en l-uplets, on peut transformer l'une en l'autre par des transpositions et des rotations.*

Preuve Les chaînes ayant une décomposition en l-uplets donnée correspondent aux chemins eulériens dans le graphe orienté G (voir figure 5.10). Soit le graphe G est eulérien, soit il contient un chemin eulérien. Notons qu'il n'existe une rotation de la chaîne correspondante que si G est eulérien. Dans ce cas, les rotations correspondent simplement à un choix du sommet initial d'un cycle eulérien.

On remplace chaque arc $a = (v, w)$ dans G par deux arêtes (non orientées) : (v, a) coloriée en blanc et (a, w) coloriée en noir (voir figure 5.10). Évidemment,

chaque chemin alterné dans le nouveau graphe G^* est un chemin orienté dans G, et vice versa. D'après le théorème 2.2, les changements d'ordre et les réflexions engendrent tous les chemins eulériens dans G^* et, par conséquent, toutes les chaînes ayant une décomposition en l-uplets donnée. Notons que les transpositions d'Ukkonen correspondent aux changements d'ordre dans G^*. D'un autre côté, tout cycle dans G^* est pair ; il n'y a donc pas de réflexion d'ordre dans G^*. Ceci prouve la conjecture d'Ukkonen selon laquelle les transpositions et les rotations engendrent toutes les chaînes ayant une décomposition en l-uplets donnée. ■



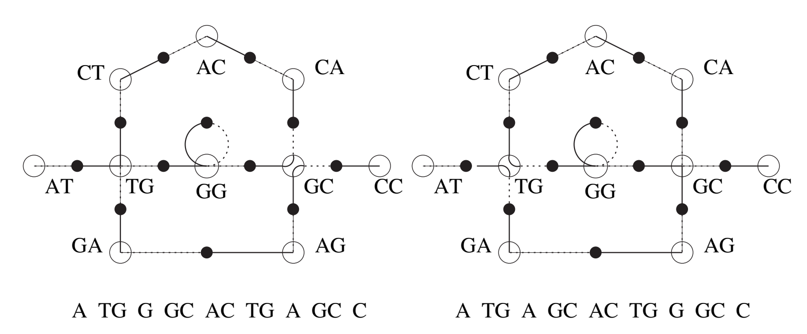


Figure 5.10 – La conjecture d'Ukkonen. La séquence (chemin eulérien) de droite est obtenue à partir de la séquence de gauche par une transposition définie sur les paires intercalées de dinucléotides TG et GC.

5.7 Cycles eulériens 2-optimaux

Un 2-*chemin* (v_1, v_2, v_3) dans un graphe orienté est un chemin constitué des arcs (v_1, v_2) et (v_2, v_3). Tout cycle eulérien dans un graphe $G(V, E)$ définit un ensemble de $|E|$ 2-chemins correspondant à chaque paire d'arcs consécutifs de ce cycle. Un ensemble de $|E|$ 2-chemins est dit *valide* si chaque arc de E apparaît deux fois dans cet ensemble : une fois au début d'un 2-chemin et une fois à la fin d'un 2-chemin. Tout ensemble valide de 2-chemins définit une décomposition en cycles de E. Un ensemble valide de 2-chemins qui définit un cycle eulérien est appelé un *ensemble d'Euler* de 2-chemins.

Soit G un graphe eulérien orienté avec un *poids* $w(v_1 v_2 v_3)$ associé à chaque 2-chemin $(v_1 v_2 v_3)$ dans G. Le *poids d'un cycle eulérien* $C = v_1 \dots v_m$ est la somme des poids de ses 2-chemins : $w(C) = \sum_{i=1}^{m-1} w(v_i v_{i+1} v_{i+2})$ (on suppose $v_m = v_1$ et $v_{m+1} = v_2$). Le *problème du cycle eulérien 2-optimal* est de trouver un cycle eulérien de poids maximal (Gusfield *et al.*, 1998 [147]).

Soit :

$$C = \dots \overline{v} x \underline{\overline{w} \dots \overline{\overline{v}}} y \overline{\overline{w}} \dots \hat{v} x \underbrace{\hat{w} \dots \hat{\hat{v}}} y \hat{\hat{w}} \dots$$

un cycle eulérien dans G qui traverse les sommets x et y dans un ordre intercalé. Un changement d'ordre qui transforme C en

$$C' = \dots \overline{v} x \underbrace{\hat{w} \dots \hat{\hat{v}}} y \overline{\overline{w}} \dots \hat{v} x \underline{\overline{w} \dots \overline{\overline{v}}} y \hat{\hat{w}} \dots$$

est appelé un *échange eulérien* de C en x, y. Pour tout couple de cycles eulériens dans un graphe orienté, on peut transformer l'un en l'autre par une suite de changements eulériens (Pevzner, 1995 [267]).

Un échange eulérien de C en x, y est noté $s = s(x, y)$ et l'on écrit $C' = s \cdot C$. Un échange eulérien s de C en x, y transforme quatre 2-chemins dans C :

$$\overline{v} x \overline{w}, \overline{\overline{v}} y \overline{\overline{w}}, \hat{v} x \hat{w}, \hat{\hat{v}} y \hat{\hat{w}},$$

en quatre 2-chemins dans C' :

$$\overline{v} x \hat{w}, \hat{\hat{v}} y \overline{\overline{w}}, \hat{v} x \overline{w}, \overline{\overline{v}} y \hat{\hat{w}}.$$

On note :

$$\Delta_x(s) = w(\overline{v} x \hat{w}) + w(\hat{v} x \overline{w}) - w(\overline{v} x \overline{w}) - w(\hat{v} x \hat{w})$$

$$\Delta_y(s) = w(\hat{\hat{v}} y \overline{\overline{w}}) + w(\overline{\overline{v}} y \hat{\hat{w}}) - w(\overline{\overline{v}} y \overline{\overline{w}}) - w(\hat{\hat{v}} y \hat{\hat{w}})$$

et $\Delta(s) = \Delta_x(s) + \Delta_y(s)$. La quantité $\Delta(s)$ représente le changement entre le poids du cycle eulérien C et celui de C'. Un échange eulérien s est *croissant* pour le cycle C s'il vérifie $w(s \cdot C) > w(C)$.

Considérons un ensemble valide de 2-chemins S contenant quatre 2-chemins :

$$\overline{v} x \overline{w}, \overline{\overline{v}} y \overline{\overline{w}}, \hat{v} x \hat{w}, \hat{\hat{v}} y \hat{\hat{w}}.$$

Un *échange* de cet ensemble en les sommets x, y est un nouvel ensemble valide de 2-chemins contenant les quatre 2-chemins suivants à la place des quatre précédents :

$$\overline{v}x\hat{w}, \hat{\hat{v}}y\overline{\overline{w}}, \hat{v}x\overline{w}, \overline{\overline{v}}y\hat{\hat{w}}.$$

Si S est un ensemble d'Euler de 2-chemins, alors un échange de S est dit non eulérien s'il transforme S en un ensemble non eulérien de 2-chemins. Par exemple, un échange en x, y d'un ensemble de 2-chemins correspondant au cycle eulérien $\ldots\overline{v}x\underline{\overline{w}\ldots\hat{v}}x\hat{w}\ldots\overline{\overline{v}}y\underbrace{\overline{\overline{w}}\ldots\hat{\hat{v}}}y\hat{\hat{w}}\ldots$ est un échange non eulérien.

Gusfield *et al.*, 1998 [147] ont étudié le problème du cycle eulérien 2-optimal dans des graphes orientés de degré maximal 2.

Théorème 5.6 *Si C et C^* sont deux cycles eulériens d'un graphe de degré maximal 2, alors on peut passer de l'un à l'autre par une suite d'échanges eulériens $s_1, \ldots s_t$ qui vérifient $\Delta(s_1) \geq \Delta(s_2) \geq \ldots \geq \Delta(s_t)$.*

Preuve Soit $s_1, \ldots, s_t$ une suite d'échanges eulériens transformant C en C^*, de telle sorte que le vecteur $(\Delta(s_1), \Delta(s_2), \ldots, \Delta(s_t))$ soit *lexicographiquement maximal* parmi toutes les suites d'échanges eulériens transformant C en C^*. Si ce vecteur ne vérifie pas la condition $\Delta(s_1) \geq \Delta(s_2) \geq \ldots \geq \Delta(s_t)$, alors il existe un indice i avec $1 \leq i \leq t-1$ et $\Delta(s_i) < \Delta(s_{i+1})$. On pose $C' = s_{i-1} \cdots s_1 C$. Si l'échange s_{i+1} est eulérien dans C' (autrement dit, l'échange du système de 2-chemins dans C' imposé par s_{i+1} définit un cycle eulérien), alors $s_1, \ldots, s_{i-1}, s_{i+1}$ est lexicographiquement supérieur à $s_1, \ldots, s_{i-1}, s_i$, ce qui est contradictoire. (Pevzner, 1995 [267] implique qu'il existe une transformation de C en C^* avec le préfixe $s_1, \ldots s_{i-1}, s_{i+1}$). Par conséquent, l'échange s_{i+1} en x, y n'est pas eulérien dans C'. Ceci implique que les occurrences de x et y dans C' ne sont pas intercalées : $C' = \ldots x \ldots x \ldots y \ldots y \ldots$ Par ailleurs, comme l'échange s_i en z, u est eulérien dans C', les occurrences de z et u dans C' sont intercalées : $C' = \ldots z \ldots u \ldots z \ldots u \ldots$ Il nous faut considérer toutes les sortes d'arrangements intercalés des quatre sommets x, y, z et u dans C'. La condition qui impose à s_{i+1} d'être un échange eulérien dans $s_i \cdot C'$ (autrement dit, s_i déplace les sommets x et y de telle sorte qu'ils s'intercalent dans $s_i \cdot C'$) rend la plupart de ces cas invalides (en fait, ceci est l'idée clé de la preuve). Il est facile de voir que tous les arrangements valides de x, y, z et u dans C' sont « équivalents » à l'arrangement :

$$C' = \ldots z\underline{\ldots x \ldots}u \ldots x \ldots y \ldots z\underbrace{\ldots}u \ldots y \ldots$$

Dans ce cas, s_i « insère » x entre les occurrences de y, intercalant ainsi x et y :

$$s_i \cdot C' = \ldots z\underbrace{\ldots}u \ldots x \ldots y \ldots z\underline{\ldots x \ldots}u \ldots y \ldots$$

Dans ce cas, les échanges s' en z, y et s'' en x, u sont également des échanges eulériens dans C' qui vérifient l'égalité $\Delta(s_i) + \Delta(s_{i+1}) = \Delta(s') + \Delta(s'')$. Sans

perte de généralité, supposons que l'on ait $\Delta(s') \geq \Delta(s'')$. On obtient alors :

$$\Delta(s') \geq \frac{\Delta(s') + \Delta(s'')}{2} = \frac{\Delta(s_i) + \Delta(s_{i+1})}{2} > \Delta(s_i).$$

Par conséquent, le vecteur $(\Delta(s_1), \ldots, \Delta(s_{i-1}), \Delta(s'))$ est lexicographiquement supérieur au vecteur $(\Delta(s_1), \ldots, \Delta(s_{i-1}), \Delta(s_i))$, ce qui amène une contradiction. ■

Le théorème 5.6 implique le résultat suivant :

Théorème 5.7 *(Gusfield* et al.*, 1998 [147]) Si C est un cycle eulérien qui n'est pas 2-optimal, alors il existe un échange eulérien croissant de C.*

La preuve du théorème 5.6 implique également que, dans le cas où tous les poids des 2-chemins sont distincts, un algorithme glouton choisissant à chaque étape l'échange de poids maximal aboutit à un cycle eulérien 2-optimal.

5.8 Séquençage positionnel par hybridation

Bien que les puces à ADN aient été proposées initialement pour le séquençage de l'ADN, leur puissance de résolution est plutôt faible. Avec des puces de 64 Kb, seuls des fragments d'ADN de 200 bp de long peuvent être reconstruits en une seule expérience de SBH. Pour améliorer la puissance de résolution du SBH, Broude *et al.*, 1994 [49] ont suggéré le *SBH positionnel* (PSBH), qui permet (avec un travail expérimental supplémentaire) de mesurer les positions approximatives de chaque l-uplet dans un fragment d'ADN cible. Bien que ceci rende la construction moins ambiguë, on ne connaît pas d'algorithme polynomial pour la reconstruction de séquences PSBH. Le PSBH peut se ramener à la recherche d'un chemin eulérien avec la restriction supplémentaire suivante : la position de tout arc dans le chemin eulérien trouvé doit être dans l'ordre des positions associées à l'arc.

Le PSBH motive le problème du *chemin eulérien positionnel*. La donnée initiale du problème du chemin eulérien positionnel est un graphe eulérien $G(V, E)$ dont chaque arc est associé à un intervalle d'entiers. Le problème est de trouver un chemin eulérien $e_1, \ldots, e_{|E|}$ dans G tel que l'intervalle de e_i contienne i :

Problème du chemin eulérien positionnel Étant donnés un multigraphe orienté $G(V, E)$ et un intervalle $I_e = \{l_e, h_e\}$, $l_e \leq h_e$, associé à chaque arc $e \in E$, trouver une chemin eulérien $e_1, \ldots, e_{|E|}$ dans G vérifiant $l_{e_i} \leq i \leq h_{e_i}$ pour $1 \leq i \leq |E|$.

Hannenhalli *et al.*, 1996 [156] ont montré que le problème du chemin eulérien positionnel est NP-complet. Ils ont présenté des algorithmes polynomiaux pour résoudre un cas particulier du PSBH, où l'intervalle des positions possibles

pour tout arc est borné par une constante (mesures expérimentales précises des positions dans le PSBH).

Steven Skiena a proposé une formulation légèrement différente du problème PSBH qui modélise les données expérimentales de façon plus adéquate. Pour cette nouvelle formulation, l'algorithme du chemin eulérien 2-optimal décrit dans la section précédente fournit une solution.

Les données expérimentales du PSBH apportent de l'information sur les positions approximatives des l-uplets, mais elles *ne fournissent* habituellement *pas* d'information sur les erreurs d'intervalles. Par suite, au lieu d'un intervalle $\{l_e, h_e\}$ associé à chaque arc, on connaît seulement la position approximative m_e qui lui est associée. Dans une formulation différente plus adéquate du problème PSBH, le but est de minimiser $\sum_{i=0}^{|E|} |(m_{e_{i+1}} - m_{e_i})|$, où $m_{e_0} = 0$ et $m_{e_{|E|+1}} = |E| + 1$. Pour toute paire d'arcs consécutifs e, e' dans G, on définit le poids du 2-chemin correspondant comme étant égal à $|m_e - m_{e'}|$. Le problème PSBH consiste trouver un chemin eulérien 2-optimal de poids minimal.

5.9 Construction de puces à ADN

Comme le nombre de caractéristiques d'une puce à ADN est fixé, on s'intéresse à la construction d'un des plus petits ensembles de sondes suffisants pour séquencer presque toutes les chaînes d'une longueur donnée. Supposons que le nombre de positions m sur une puce à ADN soit fixé et que le problème consiste à combiner m sondes pour donner la *puissance de résolution* maximale à une puce à ADN. Il s'avère que les puces *uniformes* $C(l)$ qui contiennent tous les l-uplets sont plutôt redondantes. Pevzner et Lipshutz, 1994 [271] ont introduit de nouvelles puces ayant une puissance de résolution supérieure à celle des puces uniformes. Ces puces sont fondées sur le principe de *regroupement* des sondes en *multisondes* : synthétiser un ensemble de sondes diverses à chaque adresse sur la puce. Une multisonde est un *ensemble* de sondes situées à une seule et même adresse sur la puce. Un fragment d'ADN s'hybride à une multisonde s'il s'hybride à *au moins une* sonde dans la multisonde. Par exemple, WWS est une multisonde constituée de huit sondes :

$$AAG, AAC, ATG, ATC, TAG, TAC, TTA, TTC$$

(W désigne A ou T, tandis que S remplace G ou C). RYR est une multisonde formée de huit sondes :

$$ATA, ATG, ACA, ACG, GTA, GTC, GCA, GCG$$

(R désigne les *purines* A ou G, alors que Y représente les *pyrimidines* T ou C). TXG est une multisonde constituée de quatre sondes :

$$TAG, TTG, TGG, TCG$$

(X désigne *n'importe quel* nucléotide — A, T, G ou C).

Une puce est désormais définie comme un ensemble de multisondes C, chacune d'elles étant un ensemble de sondes. La *mémoire* $|C|$ de la puce est le nombre de multisondes dans C. Chaque séquence d'ADN F définit un sous-ensemble de la puce C constitué des multisondes qui s'hybrident à F (le *spectre* de F dans C) :

$$F_C = \{p \in C : \text{ la multisonde } p \text{ contient une sonde présente dans } \overline{F}\}$$

($\overline{F}$ désigne la séquence complémentaire à F).

La *puce binaire* $C_{bin}(l)$ est la puce de mémoire $|C_{bin}(l))| = 2 \times 2^l \times 4$, qui est composée de toutes les multisondes de deux sortes :

$$\underbrace{\{W,S\},\{W,S\},\ldots,\{W,S\},}_{l}\{N\} \text{ et } \underbrace{\{R,Y\},\{R,Y\},\ldots,\{R,Y\},}_{l}\{N\},$$

où N est un nucléotide spécifié A,T, G ou C. Chaque sonde est un mélange de 2^l sondes de longueur $l+1$. Par exemple, la puce $C_{bin}(1)$ est formée des seize multisondes

$$WA, WC, WG, WT, SA, SC, SG, ST, RA, RC, RG, RT, YA, YC, YG, YT.$$

Chaque multisonde est un regroupement de deux dinucléotides (voir figure 5.11).

Puces binaires

R={A,G} Y={T,C}
W={A,T} A T
S={G,C} G C

Chaque chaîne de longueur *l* dans l'alphabet {W,S} ou {R,Y}
est un regroupement de 2^l chaînes dans l'alphabet {A,T,G,C}

{W S} . . . {W S} {A T G C}
l fois
WS - sous-puce

{R Y} . . . {R Y} {A T G C}
l fois
RY - sous-puce

Figure 5.11 – Puces binaires.

La *puce à insertions* $C_{gap}(l)$ (Pevzner *et al.*, 1991 [272]) est la puce de mémoire $|C_{gap}(l)| = 2 \times 4^l$, composée de toutes les multisondes de deux sortes :

$$N_1 N_2 \ldots N_l \text{ et } N_1 N_2 \ldots N_{l-1} \underbrace{XX \ldots X}_{l-1} N_l,$$

où N_i est une base spécifiée et X une base non spécifiée. Chaque multisonde du premier type est formée de la seule sonde de longueur l ; chaque multisonde du second type est constituée de 4^{l-1} sondes de longueur $2l - 1$.

La *puce alternante* $C_{alt}(l)$ est la puce ayant pour mémoire $|C_{alt}(l)| = 2 \times 4^l$ et qui est composée de toutes les multisondes de deux types :

$$N_1 X N_2 X \ldots N_{l-2} X N_{l-1} X N_l \quad \text{et} \quad N_1 X N_2 X \ldots N_{l-2} X N_{l-1} N_l.$$

Chaque multisonde du premier type est constituée de 4^{k-1} sondes de longueur $2k - 1$, alors que chaque multisonde de la seconde sorte est formée de 4^{k-2} sondes de longueur $2k - 2$.

5.10 Puissance de résolution des puces à ADN

Considérons la séquence $F = X_1 \ldots X_{m-1} X_m X_{m+1} \ldots X_n$ et supposons que son préfixe $F_m = X_1 X_2 \ldots X_m$ ait déjà été reconstruit. Nous allons estimer la probabilité d'étendre le préfixe de façon non ambiguë d'un nucléotide vers la droite.

Comme F_m est une reconstruction possible des m premiers nucléotides de F, on a :

$$(F_m)_C \subset F_C.$$

Il y a quatre façons d'étendre F_m : $F_m A$, $F_m T$, $F_m G$ et $F_m C$. On dit d'une extension de F_m d'un nucléotide N que c'est une *extension possible* si elle vérifie

$$(F_m N)_C \subset F_C. \tag{5.1}$$

Une séquence F est dite *extensible* après m relativement à la puce C si la condition (5.1) est vérifiée par exactement un des quatre nucléotides ; dans le cas contraire, F est dite *non extensible.*

On définit $\varepsilon(C, F, m)$ comme suit :

$$\varepsilon(C, F, m) = \begin{cases} 0, & \text{si } F \text{ est extensible après } m \text{ relativement à la puce } C \\ 1, & \text{sinon.} \end{cases}$$

La *probabilité de ramification* $p(C, n, m)$ est la probabilité qu'une n-séquence aléatoire soit non-extensible après le m-ième nucléotide par la reconstruction avec la puce C ; ainsi l'on a :

$$p(C, n, m) = \frac{1}{4^n} \sum_F \varepsilon(C, F, m),$$

où l'on somme sur les 4^n séquences F de longueur n.

Fixons m et notons $p(C,n)=p(C,n,m)$. Évidemment, $p(C,n)$ est une fonction croissante de n. Pour une probabilité p donnée, le n maximal qui vérifie la condition $p(C,n) \leq p$ est la longueur de séquence maximale $n_{max}(C,p)$ permettant une *extension non ambiguë* de probabilité de ramification inférieure à p. On démontre que, pour des puces uniformes, on a $n_{max}(C,p) \approx \frac{1}{3} \times |C| \times p$, alors que, pour des puces binaires, on obtient $n_{max}(C,p) \approx \frac{1}{\sqrt{12}} \times |C| \times \sqrt{p}$. Par conséquent, les nouvelles puces apportent un facteur d'amélioration de $\sqrt{\frac{3}{4p}}$ pour la longueur de séquence maximale, comparativement aux puces uniformes. Pour $p = 0,01$ et des puces binaires de 64 kb, on a $n_{max} \approx 1800$, contre $n_{max} \approx 210$ pour des puces uniformes.

5.11 Puces multisondes contre puces uniformes

Considérons la séquence $F = X_1 \ldots X_{m-1}X_mX_{m+1} \ldots X_n$ et supposons que son préfixe $F_m = X_1 \ldots X_m$ ait déjà été reconstruit. On note le dernier $(l-1)$-uplet de F_m sous la forme $V = X_{m-l+2} \ldots X_m$. Dans un souci de simplicité, on suppose : $l \leq m$ et $l \ll n \ll 4^l = |C(l)|$.

La séquence F est non extensible après m à l'aide de $C(l)$ si le spectre $F_{C(l)}$ contient un l-uplet VY_i (ici, Y_i désigne un nucléotide arbitraire différent de X_{m+1}). Par conséquent, on a $p(C(l),n) = 1 - P\{VY_1, VY_2, VY_3 \notin F_{C(l)}\}$. Supposons que la probabilité de trouver chacun des 4^l l-uplets en une position donnée de F soit égale à $\frac{1}{4^l}$. La probabilité que le spectre de F ne contienne pas VY_1 peut être estimée à $(1-\frac{1}{4^l})^{n-l+1}$ approximativement (en ignorant les éventuels auto-chevauchements (Pevzner *et al.*, 1989 [269]) et les effets marginaux). La probabilité que le spectre de F ne contienne ni VY_1, ni VY_2, ni VY_3 peut être estimée à $((1-\frac{1}{4^l})^{(n-l+1)})^3$. Ainsi :

$$\begin{aligned} p(C(l),n) &= 1 - P\{VY_1, VY_2, VY_3 \notin F_{C(l)}\} \\ &\approx 1 - ((1-\frac{1}{4^l})^{n-l+1})^3 \\ &\approx \frac{3n}{4^l} = \frac{3n}{|C(l)|}. \end{aligned} \tag{5.2}$$

On en déduit :

$$n_{max}(C(l),p) \approx \frac{1}{3} \times |C(l)|p.$$

Estimons maintenant $p(C_{bin}(l),n)$ pour $n \ll ||C_{bin}(l)||$. On note $V = X_{m-l+1}, \ldots, X_m$ le dernier l-uplet du préfixe F_m ; soient V_{WS} et V_{RY} son écriture dans les alphabets respectifs $\{W,S\}$ et $\{R,Y\}$. Dans ce cas, l'ambiguïté dans la reconstruction apparaît lorsque le spectre $F_{C_{bin}(l)}$ contient *à la fois* une multisonde $V_{WS}Y_i$ et une multisonde $V_{RY}Y_i$ pour $Y_i \neq X_{m+1}$. On suppose que la probabilité de trouver une multisonde provenant de $C_{bin}(l)$ en une position

donnée de F est égale à $\frac{1}{4\times 2^l}$ et on ignore les auto-chevauchements. La probabilité que le spectre de F ne contienne pas $V_{WS}Y_1$ peut donc être estimée à $(1-\frac{1}{4\times 2^l})^{n-l}$. Par conséquent, la probabilité que le spectre de F contienne à la fois $V_{WS}Y_1$ et $V_{RY}Y_1$ est :

$$(1-(1-\frac{1}{4\times 2^l})^{n-l})\times(1-(1-\frac{1}{4\times 2^l})^{n-l})\approx\frac{n^2}{4\times 2^l\times 4\times 2^l}.$$

De la même façon que pour (5.2), on en déduit :

$$\begin{array}{rcl} p(C_{bin}(l),n) & = & P\{V_{WS}Y_i\in F_{C_{bin}(l)} \text{ et } V_{RY}Y_i\in F_{C_{bin}(l)}\} \\ & \approx & 1-(1-\frac{n^2}{4\times 2^l\times 4\times 2^l})^3 \\ & \approx & 3\times\frac{n}{4\times 2^l}\times\frac{n}{4\times 2^l}=\frac{12n^2}{|C_{bin}(l)|^2}. \end{array}$$

Par conséquent, pour $C_{bin}(l)$, on a :

$$n_{max}(C_{bin}(l),p)\approx\frac{1}{\sqrt{12}}\times|C_{bin}(l)|\sqrt{p}.$$

Estimons désormais la probabilité de ramification des puces à insertions $C_{gap}(l)$. Soient $m\geq 2l-1$ et $n\ll ||C_{gap}(l)||$. Notons $U=X_{m-2l+4}\dots X_{m-l+2}$. Dans ce cas, l'ambiguïté apparaît lorsque le spectre $F_{C_{gap}(l)}$ contient une l-multisonde VY_i *et* une $(2l-1)$-multisonde $U\underbrace{XX\dots X}_{l-1}Y_i$ (ici, $Y_i\neq X_{m+1}$). On suppose que la probabilité de trouver chaque multisonde de $C_{gap}(l)$ à une position donnée de F est $\frac{1}{4^l}$ et on ignore les auto-chevauchements. La probabilité que le spectre de F ne contienne pas VY_1 peut donc être estimée à environ $(1-\frac{1}{4^l})^{n-l}$. Quant à la probabilité que le spectre de F ne contienne pas $U\underbrace{XX\dots X}_{l-1}Y_1$, elle peut être estimée à approximativement $(1-\frac{1}{4^l})^{n-(2l-1)+1}$. Par conséquent, la probabilité que le spectre de F contienne VY_1 et $U\underbrace{XX\dots X}_{l-1}Y_1$ est

$$(1-(1-\frac{1}{4^l})^{n-l})\times(1-(1-\frac{1}{4^l})^{n-2l+2})\approx\frac{n^2}{4^l\times 4^l}.$$

De la même façon que pour (5.2), on en déduit :

$$\begin{array}{rcl} p(C_{gap}(l),n) & = & P\{VY_i\in F_{C_{gap}(l)} \text{ et } UY_i\in S(C_{gap}(l),F)\} \\ & \approx & 1-(1-\frac{n^2}{4^l\times 4^l})^3 \\ & \approx & 3\times\frac{n}{4^l}\times\frac{n}{4^l}=\frac{12n^2}{|C_{gap}(l)|^2}. \end{array}$$

Des arguments similaires démontrent :

$$n_{max}(C_{alt}(l),p)\approx\frac{1}{\sqrt{12}}\times|C_{alt}(l)|\sqrt{p}.$$

5.12 Fabrication de puces à ADN

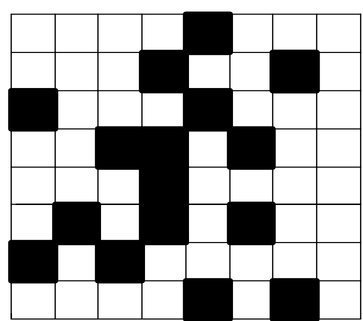
Les puces à ADN peuvent être fabriquées à l'aide de la synthèse de polymères immobilisés à très grande échelle (VLSIPS pour *very large scale immobilized polymer synthesis*)(Fodor *et al.*, 1991 [110] et Fodor *et al.*, 1993 [109]). Dans la VLSIPS, les sondes sont allongées d'un nucléotide à la fois par un procédé photolithographique qui consiste en une suite d'étapes chimiques. Chaque nucléotide porte un groupe photolabile qui protège la sonde d'autres croissances. Ce groupe peut être enlevé en inondant la sonde de lumière. À chaque étape chimique, une région prédéfinie de la puce est illuminée, ôtant ainsi un groupe de protection photolabile provenant de cette région et en « l'activant » en vue d'une croissance nucléotidique ultérieure. La puce entière est ensuite exposée à un nucléotide particulier (qui porte son propre groupe de protection photolabile), mais des réactions ne se produisent que dans la région activée. À chaque fois que l'on répète le processus, une nouvelle région est activée et un seul nucléotide est ajouté à chaque sonde dans cette région. En ajoutant des nucléotides aux régions appropriées dans la séquence appropriée, il est possible d'agrandir un ensemble complet de sondes de longueur l en seulement $4l$ étapes. La synthèse dirigée par la lumière permet d'accéder aléatoirement à toutes les positions de la puce et peut être utilisée pour faire des puces avec n'importe quelles sondes à n'importe quel site.

Les régions appropriées sont activées en illuminant la puce au travers d'une suite de masques, comme ceux de la figure 5.12. Les endroits noirs d'un masque correspondent à la région de la puce à illuminer et les blanches à celle qui doit être couverte. Malheureusement, à cause de la diffraction, de la réflexion interne et de la dispersion, les points proches du bord situés entre une région illuminée et une autre ombragée sont souvent sujets à une illumination involontaire. Dans une telle région, il n'est pas certain qu'un nucléotide sera ou non attaché. Cette incertitude engendre des sondes avec des séquences inconnues et des longueurs inconnues, qui peuvent s'hybrider à un brin d'ADN cible, compliquant ainsi l'interprétation des données expérimentales. On recherche des méthodes permettant de minimiser les longueurs de ces bords, de façon à réduire le niveau d'incertitude. Les codes bidimensionnels décrits ci-après sont des masques VLSIPS optimaux qui minimisent la longueur totale du bord de tous les masques.

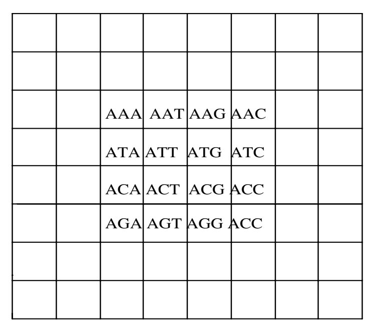
La figure 5.12 présente deux puces $C(3)$ avec différents arrangements de 3-uplets et des masques pour synthétiser le premier nucléotide A (seules les sondes ayant A pour premier nucléotide sont exhibées). La *longueur du bord* du masque situé en bas de la figure est sensiblement plus petite que celle du masque situé en haut de cette même figure. On essaie de disposer les sondes de la puce $C(l)$, de sorte que la longueur totale du bord de *tous* les $4l$ masques soit minimale. Pour deux l-uplets x et y, soit $\delta(x, y)$ le nombre de positions dans lesquelles x et y diffèrent. Il est clair que la longueur totale du bord de tous les masques vaut $2 \sum \delta(x, y)$, où l'on somme sur toutes les paires de sondes voisines

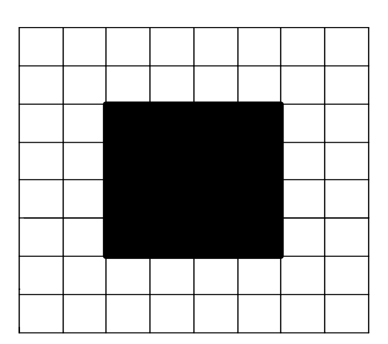
Masques pour la VLSIPS

				AGA			
			AAA			AGC	
ATG				ATT			
		ATC	ACG		AAT		
			AGT				
	ATA		AAC		AAG		
ACC		AGG					
				ACT		ACA	



longueur du bord = 58





longueur du bord = 16

Figure 5.12 – Deux masques avec des bords de longueurs différentes.

sur la puce. Cette observation établit la connexion entre la minimisation de la longueur du bord et les codes de Gray.

Un code de *Gray* de l bits est défini comme une permutation des nombres binaires entre 0 et $2^l - 1$, de telle sorte que des nombres voisins diffèrent d'exactement un bit, tout comme le premier nombre et le dernier nombre. Par exemple, le *code de Gray binaire réfléchi* de 4 bits figure ci-dessous :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

Ce code de Gray peut être produit par récurrence, en commençant avec le code de Gray de 1 bit :

$$G_1 = \{0, 1\},$$

Pour un code de Gray de l bits :

$$G_l = \{g_1, g_2, ..., g_{2^l-1}, g_{2^l}\},$$

on définit un code de Gray de $(l+1)$ bits de la façon suivante :

$$G_{l+1} = \{0g_1, 0g_2, ..., 0g_{2^l-1}, 0g_{2^l}, 1g_{2^l}, 1g_{2^l-1}, ..., 1g_2, 1g_1\}.$$

Les éléments de G_l sont simplement copiés avec des 0 en plus devant, puis renversés avec des 1 devant. On vérifie aisément que tous les éléments dans G_{l+1} sont distincts et que des éléments consécutifs dans G_{l+1} diffèrent d'exactement un bit.

On s'intéresse à un code de Gray bidimensionnel composé de chaînes de longueur l dans un alphabet à quatre lettres. En d'autres termes, on aimerait produire une matrice carrée de taille 2^l, dans laquelle chacun des 4^l l-uplets est présent à la position (i, j) et chaque paire de l-uplets adjacents (horizontalement ou verticalement) diffère d'exactement une position. Un tel code de Gray peut être produit à partir du code de Gray bidimensionnel à une lettre

$$G_1 = \begin{matrix} A & T \\ G & C \end{matrix}$$

comme suit. Pour un code de Gray bidimensionnel à l lettres

$$G_l = \begin{matrix} g_{1,1} & \cdots & g_{1,2^l} \\ \cdots & \cdots & \cdots \\ g_{2^l,1} & \cdots & g_{2^l,2^l} \end{matrix}$$

on définit le code de Gray bidimensionnel à $(l+1)$ lettres comme étant :

$$G_{l+1} = \begin{matrix} Ag_{1,1} & \cdots & Ag_{1,2^l} & Tg_{1,2^l} & \cdots & Tg_{1,1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ Ag_{2^l,1} & \cdots & Ag_{2^l,2^l} & Tg_{2^l,2^l} & \cdots & Tg_{2^l,1} \\ \\ Gg_{2^l,1} & \cdots & Gg_{2^l,2^l} & Cg_{2^l,2^l} & \cdots & Cg_{2^l,1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ Gg_{1,1} & \cdots & Gg_{1,2^l} & Cg_{1,2^l} & \cdots & Cg_{1,1} \end{matrix}$$

En particulier, on a :

$$G_2 = \begin{matrix} AA & AT & TT & TA \\ AG & AC & TC & TG \\ GG & GC & CC & CG \\ GA & GT & CT & CA \end{matrix}$$

Les éléments de G_l sont copiés dans le quadrant supérieur gauche de G_{l+1}, puis réfléchis horizontalement et verticalement dans les trois quadrants adjacents. Les A, T, C et G sont placés en face des éléments dans les quadrants supérieur gauche, supérieur droit, inférieur droit et inférieur gauche, respectivement.

La construction ci-dessus est l'un des nombreux codes de Gray possibles. Les codes de Gray bidimensionnels peuvent être produits à partir de toute paire de codes de Gray unidimensionnels G_1 et G_2 en prenant le produit bidimensionnel $G(i, j) = G_1(i) * G_2(j)$, où $*$ est une opération de *mélange* (un mélange arbitraire fixé comme celui de deux piles de cartes). Le mélange le plus simple est la concaténation de $G_1(i)$ et $G_2(j)$.

Pour des puces uniformes, les masques des codes de Gray ont des longueurs totales de bord minimales parmi tous les masques ; le quotient de la longueur du bord du masque du code de Gray sur celle du masque standard est proche de $\frac{1}{2}$ (Feldman et Pevzner, 1994 [99]).

5.13 Quelques autres problèmes et approches

5.13.1 SBH avec des bases universelles

Preparata *et al.*, 1999 [280] sont allés plus loin dans l'idée des puces à multisondes et ont décrit les puces qui atteignent la borne inférieure pour le nombre de sondes requises pour une reconstruction non ambiguë d'une chaîne arbitraire de longueur n. Ces puces utilisent des bases universelles comme l'inosine, qui s'empile correctement sans agglomérant et qui joue le rôle de symboles « joker » dans les sondes. La fabrication de ces puces est semblable à celle des puces à insertions, avec des modèles d'insertions plus élaborés. Une autre approche au regroupement dans le SBH a été proposée par Hubbell, 2000 [171].

5.13.2 SBH adaptatif

L'idée du SBH adaptatif peut être expliquée à l'aide de l'exemple suivant. Imaginons un super-programmeur qui implémente un logiciel très compliqué permettant de calculer un super-nombre. Après le premier passage sur le SuperPC, il apprend qu'un cycle dans son programme présente un goulot d'étranglement et qu'il faudra une centaine d'années avant que le super-nombre ne soit calculé. L'approche habituelle permettant de venir à bout de ce problème dans l'industrie logicielle est d'analyser le cycle chronophage, de localiser le goulot d'étranglement et d'écrire un nouveau logiciel plus rapide (ou de modifier l'ancien). Cependant, si le super-programmeur est cher (c'est le cas, de nos jours !), ceci n'est peut-être pas la meilleure approche. Une autre façon de faire serait d'analyser le cycle chronophage et de construire un *nouveau hardware* qui exécute suffisamment rapidement le cycle du goulot d'étranglement pour que l'on puisse calculer le super-nombre *avec le logiciel existant*. Évidemment, ceci n'a

de sens que si le coût de construction d'un nouveau SuperPC est inférieur au coût du super-programmeur.

Pour le SBH adaptatif, une puce à ADN tient le rôle de l'ordinateur et l'ADN celui du programme. Nous ne sommes pas libres de changer le programme (le fragment d'ADN), mais nous pouvons construire de nouvelles puces après avoir reçu des informations sur les goulots d'étranglement dans la reconstruction de séquences. Skiena et Sundaram, 1995 [316] et Margaritas et Skiena, 1995 [232] ont étudié le SBH adaptatif sans erreur et ont trouvé d'élégantes bornes théoriques pour le nombre d'étapes nécessaires à la reconstruction de séquences. Cette idée a été développée par Kruglyak, 1998 [210].

5.13.3 Séquençage shotgun de style SBH

Idury et Waterman, 1995 [175] ont suggéré d'utiliser l'approche SBH du chemin eulérien pour l'assemblage de séquences dans le séquençage traditionnel de l'ADN. L'idée est simple et élégante : on traite chaque fragment de longueur n comme $n - l + 1$ l-uplets, pour l suffisamment grand (c'est-à-dire $l = 30$). Comme la plupart des 30-uplets sont uniques dans le génome humain, cette approche aboutit à un algorithme de séquençage très efficace dans le cas de données sans erreur. Idury et Waterman, 1995 [175] ont également tenté d'adapter cet algorithme aux cas comportant des erreurs de séquençage.

5.13.4 Sondes de fidélité pour les puces à ADN

Une approche courante pour le contrôle de qualité dans la fabrication de puces à ADN est de synthétiser un petit ensemble de sondes tests qui détectent les variations dans le procédé de fabrication. Ces sondes de fidélité sont des copies identiques de la même sonde, volontairement construites lors de différentes étapes du processus de fabrication. Une cible connue s'hybride à ces sondes et les résultats de l'hybridation reflètent la qualité du procédé de fabrication. On ne veut pas seulement détecter les variations, on désire également les analyser, de façon à nous indiquer lors de quelle étape il y a eu erreur. Hubbell et Pevzner, 1999 [172] ont décrit une approche combinatoire qui construit un petit ensemble de sondes de fidélité qui détectent les variations et signalent les étapes de fabrication erronées.

Chapitre 6

Comparaison de séquences

6.1 Introduction

La mutation dans l'ADN est un processus d'évolution naturel : les erreurs de réplication de l'ADN causent des substitutions, des insertions et des suppressions de nucléotides, aboutissant à « l'édition » de textes d'ADN. La similitude entre des séquences d'ADN peut être l'indice d'une origine commune (c'est le cas de la similitude entre les gènes de la globine chez les humains et les chimpanzées) ou d'une fonction commune (c'est le cas de la similitude entre l'oncogène *ν-sys* et l'hormone de stimulation de croissance).

Établir le lien entre des gènes causant le cancer et un gène impliqué dans la croissance (Doolittle, 1983 [89] et Waterfield, 1983 [353]) a été le premier succès dans la comparaison de séquences. Les *oncogènes* sont des gènes situés dans des virus qui causent une transformation de type cancéreux des cellules infectées. L'oncogène *ν-sys* dans le *virus du sarcome simien* cause une croissance cellulaire non contrôlée et provoque le cancer chez les singes. Le *facteur de croissance* PDGF, apparemment sans rapport, est une protéine qui stimule la croissance cellulaire. Lorsque ces gènes ont été comparés, on a trouvé une similitude significative. Cette découverte a confirmé une conjecture selon laquelle le cancer peut être causé par un gène normal de croissance ayant été modifié au mauvais moment.

Levenshtein, 1966 [219] a introduit la notion de *distance d'édition* entre des chaînes ; il s'agit du nombre minimal d'opérations d'édition nécessaires pour transformer une chaîne en une autre. Les opérations d'édition sont l'insertion et la suppression d'un symbole, ainsi que la substitution d'un symbole par un autre. La plupart des algorithmes de comparaison de séquences d'ADN utilisent cet ensemble d'opérations ou un ensemble légèrement différent. Levenshtein a introduit la définition de la distance d'édition mais n'a jamais décrit d'algorithme permettant de trouver la distance d'édition entre deux chaînes. Cet algorithme a été découvert, puis redécouvert de nombreuses fois dans différentes applications allant du traitement de la parole (Vintsyuk, 1968 [347]) à

la biologie moléculaire (Needleman et Wunsch, 1970 [251]). Bien que les détails des algorithmes diffèrent légèrement en fonction des applications, ce sont tous des variantes de la programmation dynamique.

Trouver des différences (distance d'édition) entre des séquences équivaut souvent à trouver des similitudes entre celles-ci. Par exemple, si les opérations d'édition se limitent aux insertions et aux suppressions (pas de substitution), le problème de la distance d'édition est équivalent au problème du plus long sous-mot commun (LCS pour *Longest Common Subsequence*). Les mathématiciens ont commencé à s'intéresser au problème LCS bien avant la découverte de l'algorithme de programmation dynamique pour la comparaison de séquences. Des études sur le groupe symétrique ont révélé de surprenantes connexions entre la théorie de la représentation et le problème consistant à trouver la LCS entre deux permutations. Le premier algorithme pour ce problème a été décrit par Robinson, 1938 [287]. Son travail a été oublié jusque dans les années 60, lorsque Schensted, 1961 [306] et Knuth, 1970 [201] ont redécouvert les relations existant entre la LCS et les tableaux de Young.

Bien que la plupart des aspects algorithmiques de la comparaison de séquences soient captés par le problème LCS, les biologistes préfèrent utiliser les *alignements* pour la comparaison de séquences d'ADN et de protéines. L'alignement des chaînes V et W est une matrice à deux lignes dans laquelle la première (resp. la seconde) ligne contient les caractères de V (resp. W) dans l'ordre, séparés par des espaces. Le score d'un alignement est défini comme étant la somme des scores de ses colonnes. Le score d'une colonne est généralement une valeur positive pour des lettres qui coïncident et une valeur négative pour des lettres distinctes.

Dans les tout premiers articles sur l'alignement de séquences, les scientifiques ont tenté de trouver la similitude entre des chaînes *entières* V et W, c'est-à-dire ont traité la question de l'alignement global. Ceci est très significatif pour les comparaisons entre des membres de la même famille protéique, comme les *globines*, qui sont très bien préservées et ont presque la même longueur dans les organismes allant des drosophiles aux humains. Dans de nombreuses applications biologiques, le score d'alignement entre des sous-chaînes de V et W peut être supérieur au score d'alignement entre les chaînes entières. Ce problème est connu sous le nom du problème de l'*alignement local.* Par exemple, les gènes *homéobox*, qui régulent les développements embryonnaires, sont présents dans une grande variété d'espèces. Bien que les gènes homéobox soient très différents dans les diverses espèces, une de leurs régions — appelée *homéodomaine* — est hautement préservée. On peut alors se demander comment trouver ce secteur préservé et ignorer les domaines montrant très peu de similitude. Smith et Waterman, 1981 [320] ont proposé une ingénieuse variante de la programmation dynamique qui résout le problème de l'alignement local.

En 1983, il était surprenant de trouver des similitudes entre un gène à l'origine du cancer et un gène impliqué dans la croissance normale et le développement. Aujourd'hui, il serait encore plus surprenant de *ne pas trouver* de similitude entre un gène nouvellement séquencé et l'immense base de données

GenBank. Cependant, la recherche dans celle-ci n'est pas aussi facile qu'il y a vingt ans. Lorsque l'on essaie de trouver le plus proche appariement d'un gène de longueur 10^3 dans une base de données de taille 10^9, même les algorithmes quadratiques de programmation dynamique peuvent s'avérer trop lents. Une approche possible consiste à utiliser une implémentation parallèle rapide d'algorithmes d'alignement ; une autre consiste à utiliser des heuristiques rapides qui fonctionnent habituellement bien, mais dont on n'est pas certain qu'elles trouvent le plus proche appariement.

De nombreuses heuristiques pour une recherche rapide sur base de données en biologie moléculaire utilisent la même idée de *filtrage*. Le filtrage repose sur l'observation selon laquelle un bon alignement inclut habituellement de courts fragments identiques ou très similaires. On peut donc rechercher de telles sous-chaînes courtes et les utiliser comme une graine pour une analyse ultérieure. L'idée du filtrage pour une comparaison de séquences rapide remonte au début des années 70, bien avant l'invention des algorithmes populaires FASTA et BLAST. Knuth, 1973 [202] a proposé une méthode pour l'appariement de modèles avec une mésalliance fondée sur l'observation selon laquelle les chaînes qui diffèrent d'une seule erreur doivent s'apparier de façon exacte soit dans la première moitié, soit dans la seconde. Par exemple, une recherche de motifs avec une inégalité de modèles de 9-uplets peut être réduit à la recherche exacte de 4-uplets avec une extension supplémentaire des occurences de 4-uplets en occurences approximatives de 9-uplets. Ceci fournit une opportunité pour séparer par filtrage les positions n'ayant pas de 4-uplet commun, une grande partie de toutes les paires de positions. L'idée du filtrage en bio-informatique moléculaire a d'abord été décrite par Dumas et Ninio, 1982 [92], puis elle a été reprise de manière significative par Wilbur et Lipman, 1983 [368] et Lipman et Pearson, 1985 [225] dans leur algorithme FASTA. Le filtrage a ensuite été développé dans BLAST, un outil de recherche sur base de données qui domine désormais en biologie moléculaire (Altschul *et al.*, 1990 [5]).

6.2 Problème du plus long sous-mot commun

Définissons un *sous-mot commun* aux chaînes $V = v_1 \ldots v_n$ et $W = w_1 \ldots w_m$ comme étant une suite d'indices :

$$1 \leq i_1 < \ldots < i_k \leq n$$

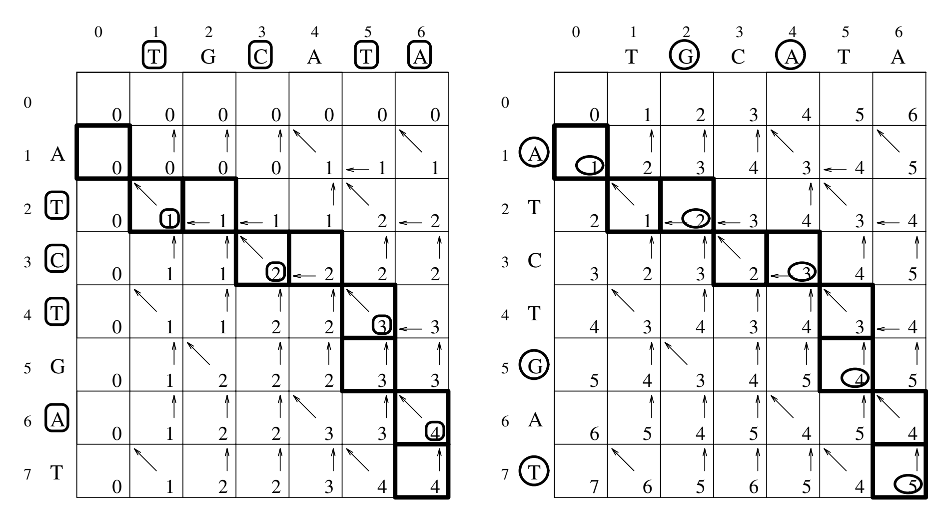
et une suite d'indices :

$$1 \leq j_1 < \ldots < j_k \leq m$$

telles que :

$$v_{i_t} = w_{j_t} \text{ pour } 1 \leq t \leq k.$$

Soit $s(V, W)$ la longueur k d'un *plus long sous-mot commun* (LCS) à V et W. Il est clair que $d(V, W) = n + m - 2s(V, W)$ est le nombre minimal d'insertions et de suppressions nécessaires pour transformer V en W. La figure 6.1 présente



Calcul de similitude s(V,W) = 4
V et W ont une sous-séquence TCTA en commun

Calcul de distance d(V,W) = 5
V peut être transformée en W en supprimant A,G,T et en insérant G,A

Alignement :
A T - C - T G A T
- T G C A T - A -

Figure 6.1 – Illustration de l'algorithme de programmation dynamique permettant de calculer le plus long sous-mot commun.

une LCS de longueur 4 pour les chaînes $V = ATCTGAT$ et $W = TGCATA$ et une plus courte séquence de deux insertions et trois suppressions transformant V en W.

Un simple algorithme de programmation dynamique calculant $s(V, W)$ a été découvert indépendemment par plusieurs auteurs. Soit $s_{i,j}$ la longueur de la LCS entre le i-préfixe $V_i = v_1 \dots v_i$ de V et le j-préfixe $W_j = w_1 \dots w_j$ de W. Soit $s_{i,0} = s_{0,j} = 0$ pour tout $1 \leq i \leq n$ et tout $1 \leq j \leq m$. On peut alors calculer $s_{i,j}$ à l'aide de la récurrence suivante :

$$s_{i,j} = \max \begin{cases} s_{i-1,j} \\ s_{i,j-1} \\ s_{i-1,j-1} + 1, & \text{si } v_i = w_j \end{cases}$$

Le premier terme (resp. le second) correspond au cas où v_i (resp. w_j) est absent de la LCS de V_i et W_j ; quant au troisième terme, il correspond au cas où v_i et w_j sont présents dans la LCS de V_i et W_j (v_i *s'apparie à* w_j). Le tableau de *programmation dynamique* de gauche de la figure 6.1 présente le calcul du score de *similitude* $s(V, W)$ entre V et W, tandis que le tableau de droite présente le calcul de la *distance d'édition* entre V et W. La distance d'édition $d(V, W)$ est calculée selon les conditions initiales $d_{i,0} = i$, $d_{0,j} = j$ pour tout $1 \leq i \leq n$

et tout $1 \leq j \leq m$ et la récurrence suivante :

$$d_{i,j} = \min \begin{cases} d_{i-1,j} + 1 \\ d_{i,j-1} + 1 \\ d_{i-1,j-1} & \text{si } v_i = w_j \end{cases}$$

La longueur d'une LCS entre V et W peut être lue à partir de l'élément (n, m) du tableau de programmation dynamique. Pour *construire* une LCS, il faut conserver l'information qui nous dit laquelle des trois quantités ($s_{i-1,j}$, $s_{i,j-1}$ ou $s_{i-1,j-1} + 1$) correspond au maximum dans la récurrence pour $s_{i,j}$ et *faire un retour arrière* à travers le tableau de programmation dynamique. L'algorithme suivant atteint ce but en utilisant les symboles $\leftarrow$, $\uparrow$ et $\nwarrow$ pour désigner les trois cas évoqués précédemment :

```
LCS (V, W)
pour i ← 1 à n
        s_{i,0} ← 0
pour i ← 1 à m
        s_{0,i} ← 0
pour i ← 1 à n
        pour j ← 1 à m
                si v_i = w_j
                        s_{i,j} ← s_{i-1,j-1} + 1
                        b_{i,j} ← ↖
                sinon si s_{i-1,j} ≥ s_{i,j-1}
                        s_{i,j} ← s_{i-1,j}
                        b_{i,j} ← ↑
                sinon
                        s_{i,j} ← s_{i,j-1}
                        b_{i,j} ← ←
retourner s et b
```

Le programme récursif qui suit imprime le plus long sous-mot commun en utilisant le retour arrière le long du chemin qui va de (n, m) à $(0, 0)$, selon les flèches $\leftarrow$, $\uparrow$ et $\nwarrow$. L'invocation initiale est **IMPRIMER-LCS**(b, V, n, m).

```
IMPRIMER-LCS (b, V, i, j)
si i = 0 ou j = 0
    retourner
si b_{i,j} = ↖
    IMPRIMER-LCS(b, V, i − 1, j − 1)
    imprimer v_i
sinon si b_{i,j} = ↑
    IMPRIMER-LCS(b, V, i − 1, j)
sinon
    IMPRIMER-LCS(b, V, i, j − 1)
```

6.3 Alignement de séquences

Soit $\mathcal{A}$ un *alphabet* à k lettres et soient V et W deux séquences sur $\mathcal{A}$. Soit $\mathcal{A}' = \mathcal{A} \cup \{-\}$ un alphabet étendu, où le symbole « $-$ » désigne l'*espace*. Un *alignement* des chaînes $V = v_1, \ldots, v_n$ et $W = w_1, \ldots, w_m$ est une matrice A de taille $2 \times l$ ($l \geq n, m$), dont la première ligne (resp. la seconde) contient les caractères de V (resp. W) dans l'ordre, entremêlés de $l-n$ (resp. $l-m$) espaces (en bas de la figure 6.1). On suppose qu'aucune colonne de la matrice d'alignement ne contient deux espaces. Les colonnes de l'alignement qui comportent un espace sont appelées des *indels* ; celles qui contiennent une espace dans la première ligne (resp. la seconde) sont appelées des *insertions* (resp. *délétions*). Les colonnes contenant la même lettre sur les deux lignes sont appelées des *appariements*, tandis que celles qui comportent des lettres différentes sont des *mésappariements*. Le score d'une colonne contenant des symboles x et y de l'alphabet étendu est défini comme étant une matrice de taille $(k+1) \times (k+1)$ qui définit les scores de similitude $\delta(x, y)$ pour chaque paire de symboles x et y de $\mathcal{A}'$. Le score d'alignement d'une matrice d'alignement est la somme des scores de ses colonnes. La fonction δ la plus simple accorde une prime $\delta(x, x) = 1$ pour les appariements et pénalise chaque mésappariement de $\delta(x, y) = -\mu$ et chaque insertion ou délétion de $\delta(x, -) = \delta(-, x) = -\sigma$. Dans ce cas, le *score* d'alignement est défini par la formule suivante : nombre d'appariements – $\mu \times$ nombre de mésappariements – $\sigma \times$ nombre d'indels. Le problème du plus long sous-mot commun correspond au problème d'alignement avec les paramètres $\mu = \infty$ et $\sigma = 0$. Les matrices communes pour la comparaison de séquences protéiques, les *matrices PAM (Point Accepted Mutation* ou *mutation acceptée pour 100 aa)* et *BLOSUM*, reflètent la fréquence avec laquelle l'acide aminé x remplace l'acide aminé y dans les séquences évolutives apparentées (Dayhoff *et al.*, 1978 [82], Altschul, 1991 [6] et Henikoff et Henikoff, 1992 [158]).

Le problème de l'*alignement de séquences* (global) consiste à trouver l'alignement de séquences V et W de score maximal. La récurrence correspondant au score $s_{i,j}$ d'alignement optimal entre V_i et W_j est

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

Chaque alignement de V et W correspond à un chemin dans le *graphe d'édition* des séquences V et W (voir figure 6.2). Par conséquent, le problème d'alignement de séquences correspond au *problème du plus long chemin* de la source au puits dans ce *graphe orienté acyclique*.

6.4 Alignement de séquences local

Des similitudes biologiquement significatives sont souvent présentes dans certaines parties des fragments d'ADN et absentes dans d'autres. Dans ce cas,

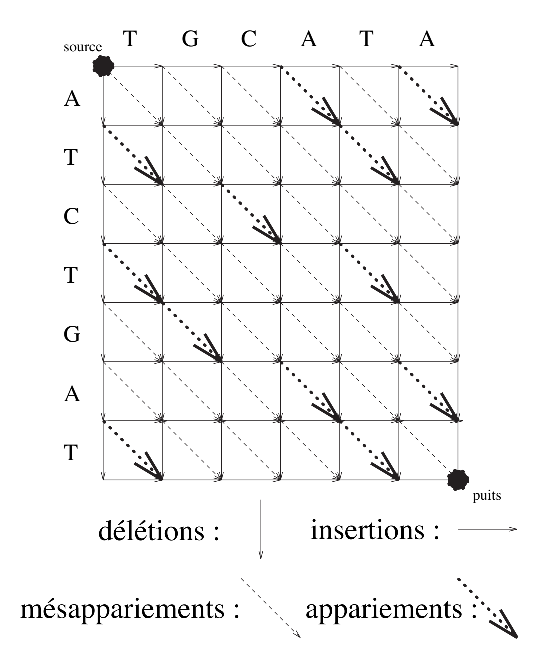


Figure 6.2 – Graphe d'édition. Les poids des arêtes d'insertion et de délétion sont égaux à $-\sigma$, ceux des arêtes de mésappariement à $-\mu$ et ceux des arêtes d'appariement sont à 1.

les biologistes tentent de maximiser $s(v_i \ldots v_{i'}, w_j \ldots w_{j'})$, où le maximum est pris sur toutes les sous-chaînes $v_i \ldots v_{i'}$ de V et $w_j \ldots w_{j'}$ de W.

Le problème de l'alignement global consiste à trouver le plus long chemin entre les sommets $(0, 0)$ et (n, m) dans le graphe d'édition, tandis que le problème de l'alignement local revient à trouver le plus long chemin parmi tous les chemins reliant deux sommets arbitraires (i, j) et (i', j') dans le graphe d'édition. Une approche directe et inefficace de ce problème consiste à trouver le plus long chemin entre toute paire de sommets (i, j) et (i', j'). Plutôt que de trouver le plus long chemin à partir de tout sommet (i, j), le problème de l'alignement local peut se restreindre au suivant : trouver les plus longs chemins qui partent de la source en ajoutant des arêtes de poids nul reliant la source à chaque autre sommet. Ces arêtes fournissent un saut « gratuit » de la source à tout autre sommet (i, j). Une légère différence dans la récurrence suivante reflète cette transformation du graphe d'édition (Smith et Waterman, 1981 [320]) :

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + \delta(v_i, -) \\ s_{i,j-1} + \delta(-, w_j) \\ s_{i-1,j-1} + \delta(v_i, w_j) \end{cases}$$

La plus grande valeur de $s_{i,j}$ représente le score de l'alignement local de V et W (au lieu de $s_{n,m}$ pour l'alignement global).

L'alignement local optimal fournit seulement le plus long chemin dans le graphe d'édition. Dans le même temps, plusieurs alignements locaux peuvent avoir une signification biologique et l'on cherche des méthodes permettant de trouver les k meilleurs alignements locaux non chevauchants (Waterman et Eggert, 1987 [359] et Huang *et al.*, 1990 [168]). Ces méthodes sont particulièrement importantes pour comparer à une autre protéine des protéines multi-domaines partageant plusieurs blocs semblables dispersés en une protéine. Dans ce cas, il n'existe pas d'alignement local simple représentant toutes les similitudes significatives.

6.5 Alignement avec pénalité de brèche

Les mutations sont habituellement des manifestations d'erreurs dans la réplication de l'ADN. La nature supprime ou insère fréquemment des sous-chaînes entières comme une unité, plutôt que de recourir à la délétion ou à l'insertion de nucléotides individuels. Une *brèche* dans l'alignement est définie comme étant une suite continue d'espaces dans l'une des lignes. Il est naturel de supposer que le score d'une brèche constituée de x espaces n'est pas simplement la somme des scores de x indels, mais plutôt une fonction plus générale. Pour des *pénalités de brèche affines*, le score d'une brèche de longueur x est de $-(\rho + \sigma x)$, où $\rho > 0$ est la pénalité de l'introduction de la brèche et $\sigma > 0$ celle de chaque symbole de la brèche. On peut s'accomoder des pénalités de brèche affines en introduisant de longues arêtes verticales et horizontales dans le graphe d'édition (par exemple, une arête allant de (i, j) à $(i + x, j)$ de longueur $-(\rho + \sigma x)$) et en calculant ultérieurement le plus long chemin dans ce graphe. Comme le nombre d'arêtes dans le graphe d'édition pour les pénalités de brèche affines augmente, il semblerait, à première vue, que la complexité temporelle de l'algorithme d'alignement croisse également de $O(n^2)$ à $O(n^3)$. Cependant, les trois récurrences suivantes maintiennent la complexité à un niveau relativement bas :

$$\overset{\downarrow}{s}_{i,j} = \max \begin{cases} \overset{\downarrow}{s}_{i-1,j} - \sigma \\ s_{i-1,j} - (\rho + \sigma) \end{cases}$$

$$\vec{s}_{i,j} = \max \begin{cases} \vec{s}_{i,j-1} - \sigma \\ s_{i,j-1} - (\rho + \sigma) \end{cases}$$

$$s_{i,j} = \max \begin{cases} s_{i-1,j-1} + \delta(v_i, w_j) \\ \overset{\downarrow}{s}_{i,j} \\ \vec{s}_{i,j} \end{cases}$$

La variable $\overset{\downarrow}{s}_{i,j}$ calcule le score de l'alignement entre V_i et W_j qui se termine par une délétion (c'est-à-dire une brèche dans W), tandis que la variable $\vec{s}_{i,j}$

calcule le score de l'alignement finissant par une insertion (c'est-à-dire une brèche dans V). Le premier terme dans les récurrences pour $\overset{\downarrow}{s}_{i,j}$ et $\vec{s}_{i,j}$ correspond à l'extension de la brèche, alors que le second correspond à l'initiation de celle-ci. Bien que les pénalités de brèche affines constituent aujourd'hui le modèle le plus utilisé, des études indiquent que les pénalités de brèche non linéaires peuvent avoir certains avantages comparées au cas affine (Waterman, 1984 [356] et Gonnet *et al.*, 1992 [133]). Des algorithmes efficaces pour l'alignement avec pénalités de brèche non linéaires ont été proposés par Miller et Myers, 1988 [236] et Galil et Giancarlo, 1989 [116].

6.6 Alignement de séquences efficace en espace

Pour la comparaison des longs fragments d'ADN, la ressource qui se trouve en quantité limitée dans l'alignement de séquences n'est pas le temps mais l'espace. Hirschberg, 1975 [163] a proposé une approche fondée sur le paradigme *diviser pour régner* et qui réalise un alignement dans un espace linéaire aux seuls dépens du doublement du temps de calcul.

La complexité temporelle de l'algorithme de programmation dynamique pour l'alignement de séquences correspond à peu près au nombre d'arêtes dans le graphe d'édition, c'est-à-dire $O(nm)$. La complexité spatiale vaut environ le nombre de sommets de ce même graphe, autrement dit $O(nm)$. Cependant, si l'on veut seulement calculer le score de l'alignement (plutôt que l'alignement lui-même), alors l'espace peut se réduire à seulement deux fois le nombre de sommets dans une seule colonne du graphe d'édition, soit $O(n)$. Cette réduction provient de l'observation selon laquelle les seules valeurs nécessaires au calcul des scores d'alignement $s_{*,j}$ (colonne j) sont les scores d'alignement $s_{*,j-1}$ (colonne $j-1$). Par conséquent, les scores d'alignement dans les colonnes situées avant $j-1$ peuvent être mis de côté durant le calcul des scores d'alignement des colonnes $j, j+1, \ldots$ En revanche, calculer l'alignement (c'est-à-dire trouver le plus long chemin dans le graphe d'édition) nécessite un retour arrière à travers la matrice entière $(s_{i,j})$. Par conséquent, celle-ci doit être stockée, conduisant à un besoin spatial en $O(nm)$.

Le plus long chemin dans le graphe d'édition relie le sommet *source* $(0, 0)$ au sommet *puits* (n, m) et passe par un *sommet médian* (inconnu) $(i, \frac{m}{2})$ (pour plus de simplicité, supposons que m soit pair). Essayons de trouver son sommet médian plutôt que de tenter de trouver l'intégralité du plus long chemin. Ceci peut être réalisé en un espace linéaire en calculant les scores $s_{*,\frac{m}{2}}$ (les longueurs des plus longs chemins allant de $(0, 0)$ à $(i, \frac{m}{2})$, pour $0 \leq i \leq n$) et les scores des chemins allant de $(i, \frac{m}{2})$ à (n, m). Ceux-ci peuvent être calculés comme étant les scores des chemins $s^{inverse}_{*,\frac{m}{2}}$ allant de (n, m) à $(i, \frac{m}{2})$ dans le graphe d'édition inverse (c'est-à-dire le graphe dont l'orientation de toutes les arêtes est inversée). La valeur $s_{i,\frac{m}{2}} + s^{inverse}_{i,\frac{m}{2}}$ est la longueur du plus long chemin allant de $(0, 0)$ à (n, m) qui passe par le sommet $(i, \frac{m}{2})$. Par conséquent,

$\max_i(s_{i,\frac{m}{2}} + s^{inverse}_{i,\frac{m}{2}})$ calcule la longueur du plus long chemin et identifie son sommet médian.

Le calcul de ces valeurs nécessite un temps égal à l'aire du rectangle de gauche (de la colonne 1 à $\frac{m}{2}$) plus l'aire du rectangle de droite (de la colonne $\frac{m}{2}+1$ à m) et un espace en $O(n)$ (voir figure 6.3). Après avoir trouvé le sommet médian $(i, \frac{m}{2})$, le problème qui consiste à découvrir le plus long chemin de $(0,0)$ à (n,m) peut se diviser en deux sous-problèmes : trouver le plus long chemin de $(0,0)$ au sommet médian $(i, \frac{m}{2})$ et trouver le plus long chemin allant de $(i, \frac{m}{2})$ à (n,m). Plutôt que d'essayer de découvrir ces chemins, on tente d'abord de trouver les sommets médians dans les rectangles correspondants (voir figure 6.3). Ceci peut être réalisé en un temps égal à l'aire de ces rectangles, qui est deux fois plus petite que la surface du rectangle original. En calculant de cette façon, on trouvera les sommets médians de tous les rectangles en un temps $aire + \frac{aire}{2} + \frac{aire}{4} + \ldots \leq 2 \times aire$ et, par conséquent, on calculera le plus long chemin en un temps $O(nm)$ et en un espace $O(n)$:

Chemin $(source, puits)$
si *source* et *puits* sont dans des colonnes consécutives
    retourner le plus long chemin allant de *source* à *puits*
sinon
    $median \leftarrow$ sommet médian entre *source* et *puits*
    Chemin $(source, median)$
    Chemin $(median, puits)$

6.7 Tableaux de Young

Une *sous-séquence croissante* d'une permutation $\pi = x_1x_2\ldots x_n$ est une suite d'indices $1 \leq i_1 < \ldots < i_k \leq n$ qui vérifient $x_{i_1} < x_{i_2} \ldots < x_{i_k}$. On définit de la même façon une sous-séquence décroissante. Trouver *la plus longue sous-séquence croissante* (LIS pour *Longest Increasing Subsequence*) équivaut à trouver la LCS entre π et la permutation identité $1\,2\ldots n$. Il est bien connu que toute permutation de n éléments possède soit une sous-séquence croissante, soit une sous-séquence décroissante de longueur au moins égale à $\sqrt{n}$. Ce résultat est étroitement lié à l'approche de programmation non dynamique de la LCS décrite dans la suite.

Une *partition* de l'entier n est une suite d'entiers positifs $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_l$ qui vérifient $\sum_{i=1}^{i=l} \lambda_i = n$. Si $\lambda = (\lambda_1\lambda_2\ldots\lambda_l)$ est une partition de n, on note $\lambda \vdash n$. Supposons $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_l) \vdash n$. Le *diagramme de Young* (ou la *forme*) λ est alors un tableau de n cellules contenant l lignes alignées à gauche, dans lequel la ligne i comporte λ_i cellules pour $1 \leq i \leq l$. Le diagramme situé à l'extrême gauche dans la figure 6.4 présente le diagramme de Young pour $(4,2,1,1) \vdash 8$, tandis que celui qui se trouve le plus à droite montre le diagramme de Young pour $(4,2,2,1) \vdash 9$. Un *tableau de Young* (ou, plus simplement, un *tableau* de forme λ) est un tableau obtenu en remplaçant les cellules

Alignement de séquences en un espace linéaire

Figure 6.3 – Alignement de séquences efficace au niveau spatial. Le temps de calcul (aire des rectangles grisés) décroît d'un facteur 2 à chaque itération.

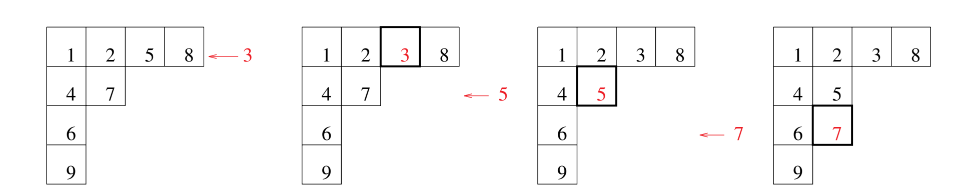


Figure 6.4 – Insertion de ligne de 3 dans des tableaux de Young.

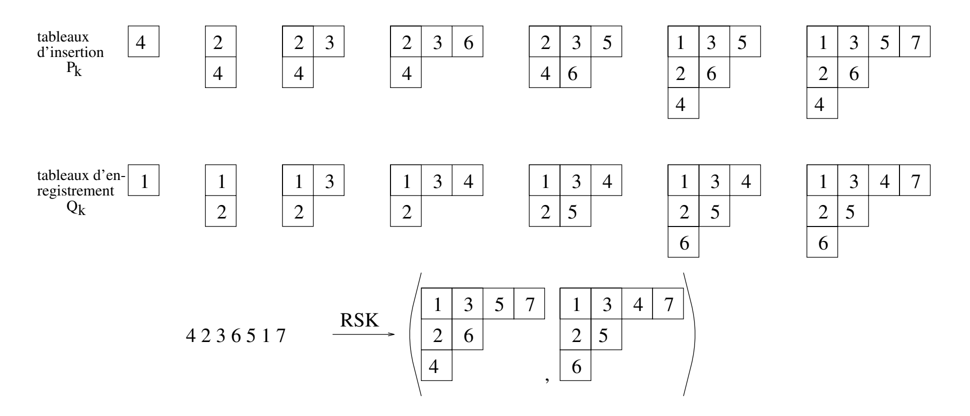


Figure 6.5 – Algorithme RSK pour la permutation $\pi = 4236517$.

du diagramme de Young λ par les nombres $1, 2, \ldots, n$ de façon bijective (tableau situé à l'extrême droite dans la figure 6.4). Un tableau λ est dit *standard* si ses lignes et ses colonnes sont des séquences croissantes. Sagan, 1991 [292] est une excellente introduction à la combinatoire des tableaux de Young et notre présentation s'inspire de cet ouvrage.

Un *bitableau* est une paire de tableaux de Young standards ayant le même diagramme de Young λ. L'algorithme Robinson-Schensted-Knuth (RSK) décrit une bijection explicite entre les bitableaux à n cellules et le groupe symétrique S_n (formé de toutes les permutations d'ordre n). Il a d'abord été proposé (en des termes plutôt flous) par Robinson, 1938 [287], en relation avec la théorie de la représentation. Schensted, 1961 [306] a ensuite redécouvert et décrit clairement cet algorithme en termes de combinatoire pure. Par la suite, Knuth, 1970 [201] a généralisé celui-ci au cas de la LCS.

Soit P un *tableau partiel*, c'est-à-dire un diagramme de Young avec des entrées distinctes, dont les lignes et les colonnes croissent. Pour un tableau R et un élément x, on définit R_{x+} comme étant le plus petit élément de R supérieur à x et R_{x-} comme le plus grand élément de R inférieur à x. Pour un x qui n'est pas dans P, on définit l'*insertion de ligne* de x dans P par l'algorithme suivant :

$R \leftarrow$ la première ligne de P
Tant que x est inférieur à un élément de la ligne R
  Remplacer R_{x+} par x dans R.
  $x \leftarrow R_{x+}$
  $R \leftarrow$ ligne suivante vers le bas.
Placer x à la fin de la ligne R.

Le résultat de l'insertion de ligne de x dans P est noté $r_x(P)$. Il faut remarquer que les règles d'insertion assurent que $r_x(P)$ possède toujours des lignes et des colonnes croissantes (voir l'exemple de la figure 6.4).

On note $\pi \overset{RSK}{\rightarrow} (P, Q)$ la bijection existant entre les permutations et les bitableaux, où $\pi \in S_n$ et P, Q sont des λ-tableaux standards, $\lambda \vdash n$. Pour une permutation $\pi = x_1 \dots x_n$, on construit une suite de tableaux

$$(P_0, Q_0) = (\emptyset, \emptyset), (P_1, Q_1), \dots, (P_n, Q_n) = (P, Q),$$

où $x_1, \dots, x_n$ sont *insérés* dans les tableaux P et $1, \dots, n$ sont *placés* dans les Q_i, de telle sorte que la forme de P_i coïncide avec celle de Q_i pour tout i.

Le placement d'un élément dans un tableau est même plus facile que l'insertion. Supposons que Q soit un tableau partiel et que (i, j) soit un coin externe de Q. Si k est supérieur à tout élément de Q, alors *on place* k dans Q à la cellule (i, j) et on pose simplement $Q_{i,j} = k$ (voir figure 6.5).

Enfin, on construit la suite de bitableaux (P_i, Q_i) de $\pi = x_1 \dots x_n$ par récurrence. En supposant que (P_{k-1}, Q_{k-1}) ait déjà été construit, on définit $P_k = r_{x_k}(P_{k-1})$ et Q_k comme le résultat du placement de k dans Q_{k-1} à la cellule (i, j), où l'insertion se termine (voir figure 6.5). On appelle P le tableau d'*insertion* et Q le tableau d'*enregistrement*.

Étant donnés le tableau $R_x(P)$ situé le plus à droite dans la figure 6.4 et la position du dernier élément ajouté (7), est-il possible de reconstruire le tableau P à l'extrême gauche de la figure 6.4 ? Comme l'élément 7 avait buté sur $R_{7-} = 5$ dans la ligne précédente (voir l'algorithme RSK), on peut reconstruire le second tableau de la figure 6.4. Comme l'élément 5 avait buté sur l'élément $R_{5-} = 3$ de la première ligne, il est également possible de reconstruire le tableau original P. Cette observation implique le théorème RSK :

Théorème 6.1 *L'application $\pi \overset{RSK}{\rightarrow} (P, Q)$ est une bijection entre les éléments de S_n et les paires de tableaux standards de même forme $\lambda \vdash n$.*

Preuve Construisons une bijection inverse $(P, Q) \overset{RSK}{\rightarrow} \pi$ en renversant l'algorithme RSK étape par étape. On commence par définir $(P_n, Q_n) = (P, Q)$. En supposant que (P_k, Q_k) ait été construit, on trouvera x_k (le k-ième élément de π) et (P_{k-1}, Q_{k-1}). Trouvons la cellule (i, j) contenant k dans Q_k. Comme il s'agit du plus grand élément de Q_k, $P_{i,j}$ doit avoir été le dernier élément déplacé dans la construction de P_k. On peut utiliser la procédure suivante pour *supprimer* $P_{i,j}$ de P. Pour plus de commodité, on suppose l'existence d'une ligne vide d'indice 0 au-dessus de la première ligne de P_k.

Poser $x \leftarrow P_{i,j}$ et effacer $P_{i,j}$
$R \leftarrow$ la $(i-1)$-ième ligne de P_k
Tant que R n'est pas la zéro-ième ligne de P_k
  Remplacer R_{x-} par x dans R.
  $x \leftarrow R_{x-}$
  $R \leftarrow$ ligne suivante vers le haut.
$x_k \leftarrow x$.

Une fois le processus de suppression que l'on vient de décrire terminé, il est facile de voir que P_{k-1} est P_k et que Q_{k-1} est Q_k avec l'élément k en moins. En continuant de cette façon, on traite en fin de compte tous les éléments de π dans l'ordre inverse. ■

Lemme 6.1 *Soit $\pi = x_1 \dots x_n$. Si x_k entre dans P_{k-1} à la colonne j, alors la plus longue sous-séquence croissante de π se terminant en x_k est de longueur j.*

Preuve On travaille par récurrence sur k. Le résultat est trivial pour $k = 1$, supposons donc qu'il soit vrai pour toutes les valeurs jusqu'à $k - 1$.

On a d'abord besoin de montrer l'existence d'une sous-séquence croissante de longueur j se terminant en x_k. Soit y l'élément de P_{k-1} situé dans la cellule $(1, j-1)$. Alors on a $y < x_k$, car x_k entre dans la colonne j. En outre, d'après l'hypothèse de récurrence, il existe une sous-séquence croissante de longueur $j - 1$ qui finit en y. En combinant cette sous-séquence avec x_k, on obtient la sous-séquence voulue de longueur j.

On doit maintenant prouver qu'il ne peut y avoir une sous-séquence plus longue finissant en x_k. Raisonnons par l'absurde et supposons qu'une telle sous-séquence existe ; on note alors x_i l'élément qui précède x_k. Par récurrence, lorsque x_i est inséré, il entre dans une colonne située à droite de la colonne j. L'élément y dans la cellule $(1, j)$ de P_i vérifie donc $y \leq x_i < x_k$. Cependant, les entrées dans une position donné d'un tableau ne sont jamais croissantes avec des insertions correspondantes (voir l'algorithme RSK). L'élément se trouvant dans la cellule $(1, j)$ de P_{k-1} est donc inférieur à x_k, ce qui contredit le fait que x_k le déplace. ■

Ce lemme implique le résultat suivant :

Théorème 6.2 *La longueur de la plus longue sous-séquence croissante de la permutation π est la longueur de la première ligne de $P(\pi)$.*

6.8 Longueur moyenne des plus longues sous-séquences communes

Soient $\mathcal{V}$ et $\mathcal{W}$ deux ensembles de chaînes de n lettres sur le même alphabet. Étant donnés $\mathcal{V}$, $\mathcal{W}$ et une mesure de probabilité p sur $\mathcal{V} \times \mathcal{W}$, on s'intéresse à la *longueur moyenne* de la LCS, qui est :

$$s(n) = \sum_{V \in \mathcal{V}, W \in \mathcal{W}} s(V, W) \times p(V, W),$$

où $s(V, W)$ est la longueur de la LCS entre V et W. Deux exemples du problème de la longueur moyenne de la LCS présentent un intérêt particulier.

La plus longue sous-séquence croissante dans la permutation aléatoire $s_{per}(n)$. L'ensemble $\mathcal{V}$ ne contient que la chaîne $(1, \ldots, n)$, $\mathcal{W}$ contient toutes les permutations de longueur n et $p(V, W) = \frac{1}{n!}$. Le problème consistant à trouver $s_{per}(n)$ a été soulevé par Ulam, 1961 [341]. Hammersley, 1972 [150] a prouvé l'égalité

$$\lim_{n \to \infty} \frac{s_{per}(n)}{\sqrt{n}} = s_{per},$$

où s_{per} est une constante. Même avant d'avoir prouvé la convergence, Baer et Brock, 1968 [15] avaient conjecturé que s_{per} valait 2, sur la base de calculs extensifs. Hammersley, 1972 [150] a prouvé : $\frac{\pi}{2} \leq s_{per} \leq e$. Par la suite, Kingman, 1973 [198] a amélioré les bornes en prouvant l'encadrement : $1,59 < s_{per} < 2,49$. Logan et Shepp, 1977 [227] et Vershik et Kerov, 1977 [342] ont prouvé l'égalité $s_{per} = 2$ en menant une analyse asymptotique techniquement difficile des tableaux de Young aléatoires et en utilisant les théorèmes 6.1 et 6.2.

La plus longue sous-séquence commune $s_k(n)$ *dans un alphabet à* k *lettres.* Les deux ensembles $\mathcal{V}$ et $\mathcal{W}$ contiennent tous les k^n mots à n lettres dans un alphabet à k lettres et $p(V, W) = \frac{1}{k^n \times k^n}$. Chvatal et Sankoff, 1975 [70] ont noté la chose suivante :

$$\lim_{n \to \infty} \frac{s_k(n)}{n} = s_k,$$

où s_k est une constante. Ils ont donné les bornes inférieure et supérieure pour s_k, qui ont été améliorées par la suite par Deken, 1979, 1983 [83, 84] et Chvatal et Sankoff, 1983 [71].

Durant les années 80, deux conjectures ont été émises concernant s_k :

Conjecture de Sankoff-Mainville [305] : $\lim_{k \to \infty}(s_k \times \sqrt{k}) = 2$.

Conjecture d'Arratia-Steele [328] : $s_k = \frac{2}{1+\sqrt{k}}$.

Ces conjectures peuvent être formulées sous la forme d'énoncés sur la longueur de la première ligne des tableaux de Young. Au lieu de trouver la longueur de la première ligne, on peut essayer de découvrir une forme limite aux tableaux de Young, qui présenterait simultanément toutes leurs caractéristiques, particulièrement la longueur de la première ligne. À première vue, ce problème semble plus général et plus difficile que de trouver simplement la longueur de la première ligne. Cependant, la preuve de $s_{per} = 2$ révèle une situation paradoxale : il peut être plus facile de trouver une forme limite aux tableaux de Young entiers que de trouver la longueur attendue d'une ligne particulière en utilisant des arguments combinatoires/probabilistes *ad hoc*. En particulier, on ne connaît pas encore de solution combinatoire *ad hoc* au problème d'Ulam.

Récemment, il y a eu une activité intense dans l'étude des plus longues sous-séquences croissantes. Aldous et Diaconis, 1995 [2] ont utilisé une représentation de particules qui interagissent (modélisée par la LIS dans une séquence de n nombres réels indépendants aléatoires distribués uniformément dans un

intervalle) pour donner une preuve différente de l'égalité $s_{per} = 2$. Baik *et al.*, 1999 [21] ont prouvé : $s_{per}(n) = 2\sqrt{n} - \mu n^{1/6} + o(n^{1/6})$, où $\mu = 1,711...$

Soit Λ_n l'ensemble de toutes les formes à n cellules. On note $P(\pi)$ un tableau d'insertion P correspondant à une permutation π et l'on considère l'ensemble des permutations $\Gamma = \{\pi : P(\pi)$ contient n dans la première ligne$\}$. Pour une forme λ donnée avec n cellules, on pose $\Gamma_\lambda = \{\pi : P(\pi)$ contient n dans la première ligne et $P(\pi)$ a la forme λ $\}$. Étant donnée une permutation aléatoire $\pi \in S_n$, soit p_n la probabilité que $P(\pi)$ contienne n dans la première ligne.

Lemme 6.2 $\forall n \in \mathbb{N}, p_n \leq \frac{1}{\sqrt{n}}$.

Preuve Si λ est une forme, on note λ_+ la forme obtenue à partir de λ en ajoutant une nouvelle cellule à la fin de la première ligne. On observe que le nombre de tableaux standards de forme $\mu \in \Lambda_n$, avec n dans la première ligne, est égal au nombre de tableaux standards de forme $\lambda \in \Lambda_{n-1}$, où $\lambda_+ = \mu$. Soit f_λ le nombre de tableaux de Young de forme λ. Selon le théorème RSK, on a :

$$|\Gamma| = \sum_{\mu \in \Lambda_n} |\Gamma_\mu| = \sum_{\lambda \in \Lambda_{n-1}} f_\lambda \times f_{\lambda_+},$$

où f_λ est le nombre de tableaux de Young de forme λ. Ceci implique les égalités suivantes :

$$\begin{aligned} p_n &= \frac{|\Gamma|}{n!} \\ &= \sum_{\lambda \in \Lambda_{n-1}} \frac{f_\lambda \times f_{\lambda_+}}{n!} \\ &= \sum_{\lambda \in \Lambda_{n-1}} \frac{(n-1)!}{n!} \times \frac{f_\lambda \times f_{\lambda_+}}{f_\lambda \times f_\lambda} \times \frac{f_\lambda \times f_\lambda}{(n-1)!} \\ &= \sum_{\lambda \in \Lambda_{n-1}} \frac{1}{n} \times \frac{f_{\lambda_+}}{f_\lambda} \times p(\lambda) \end{aligned}$$

D'après le théorème RSK, $p(\lambda) = \frac{f_\lambda \times f_\lambda}{(n-1)!}$ est la probabilité qu'une permutation aléatoire $\pi \in S_{n-1}$ corresponde à une forme λ. En notant $E(X) = p_n$ la valeur moyenne de la variable aléatoire $X = \frac{1}{n} \times \frac{f_{\lambda_+}}{f_\lambda}$, puis en appliquant l'inégalité $E(X) \leq \sqrt{E(X^2)}$, on en déduit :

$$p_n^2 \leq \sum_{\lambda \in \Lambda_{n-1}} \frac{1}{n \times n} \times \frac{f_{\lambda_+} \times f_{\lambda_+}}{f_\lambda \times f_\lambda} \frac{f_\lambda \times f_\lambda}{(n-1)!} = \sum_{\lambda \in \Lambda_{n-1}} \frac{1}{n \times n!} f_{\lambda_+} \times f_{\lambda_+} =$$

$$\frac{1}{n} \sum_{\lambda \in \Lambda_{n-1}} \frac{f_{\lambda_+} \times f_{\lambda_+}}{n!} = \frac{1}{n} \sum_{\lambda \in \Lambda_{n-1}} p(\lambda_+) \leq \frac{1}{n} \sum_{\mu \in \Lambda_n} p(\mu) \leq \frac{1}{n}$$

car λ_+ parcourt tous les $\mu \in \Lambda_n$ dont la première ligne est plus longue que la deuxième. ■

Le théorème suivant a été prouvé (et apparemment jamais publié) par Vershik et Kerov à la fin des années 70. La première preuve publiée est apparue beaucoup plus tard (Pilpel, 1990 [276]).

Théorème 6.3 $s_{per} \leq 2$.

Preuve Étant donnée $\pi \in S_n$, soit $p_k(n)$ la probabilité que l'élément k apparaisse dans la première ligne de $P(\pi)$. Notons que $p_k(n) = p_k$ (les éléments $1, \ldots, k$ d'une permutation aléatoire dans S_n sont également distribués sur tous les ordonnancements relatifs possibles). D'après le lemme 6.2, la longueur attendue de la première ligne de $P(\pi)$ est

$$r_1 = \sum_{k=1}^{n} p_k(n) = \sum_{k=1}^{n} p_k \leq \sum_{k=1}^{n} \frac{1}{\sqrt{k}}.$$

Comme on sait que $\frac{1}{\sqrt{k}} \leq 2(\sqrt{k} - \sqrt{k-1})$, on en déduit que : $r_1 \leq 2\sqrt{n}$. Comme la longueur de la première ligne de $P(\pi)$ est égale à la longueur de la plus longue sous-séquence croissante de π, on obtient : $s_{per} \leq 2$. ■

6.9 Alignement de séquences généralisé et dualité

Un *ensemble partiellement ordonné* est un couple $(P, \prec)$ où P est un ensemble et $\prec$ une *relation binaire transitive, antisymétrique et réflexive* sur P. Une *chaîne* $p_1 \prec p_2 \ldots \prec p_t$ est un sous-ensemble de P dans lequel toute paire d'éléments est comparable ; une *antichaîne* est un sous-ensemble dans lequel il n'existe pas deux éléments comparables. On dit que des ordres partiels $\prec$ et $\prec^*$ sont *conjugués* si, pour tout couple d'éléments distincts $(p_1, p_2) \in P^2$, la condition suivante est vérifiée :

$$p_1 \text{ et } p_2 \text{ sont } \prec\text{-comparables} \iff p_1 \text{ et } p_2 \text{ sont } \prec^*\text{-incomparables.}$$

On s'intéresse aux *plus longues $\prec$-séquences*, c'est-à-dire aux chaînes de longueur maximale pour $\prec$ (*alignement de séquences généralisé*). Soient $I = \{1, 2 \ldots, n\}$, $J = \{1, 2 \ldots, m\}$ et $P \subset I \times J$. On s'intéresse à la comparaison des deux séquences $V = v_1 v_2 \ldots v_n$ et $W = w_1 w_2 \ldots w_m$, avec $P = \{(i,j) : v_i = w_j\}$. Soient $p_1 = (i_1, j_1)$ et $p_2 = (i_2, j_2)$ deux éléments arbitraires dans $I \times J$. On note

$$\boldsymbol{\Delta}(p_1, p_2) = (\Delta i, \Delta j) = (i_2 - i_1, j_2 - j_1).$$

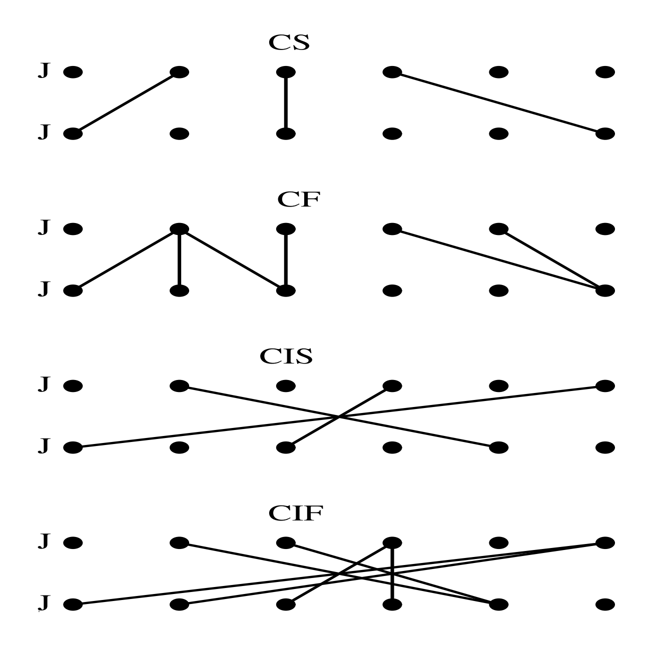


Figure 6.6 – Chaînes dans les ordres partiels CS, CF, CIS et CIF.

Considérons quelques exemples d'ordres partiels sur $I \times J$ (les chaînes correspondantes sont montrées en figure 6.6). Les ordres partiels $\prec_1$ et $\prec_4$, tout comme les ordres partiels $\prec_2$ et $\prec_3$, sont conjugués.

- Sous-séquences communes (CS pour *common subsequences*) :

$$p_1 \prec_1 p_2 \Leftrightarrow \Delta i > 0, \Delta j > 0$$

- Forêts communes (CF pour *common forests*) :

$$p_1 \prec_2 p_2 \Leftrightarrow \Delta i \geq 0, \Delta j \geq 0$$

- Sous-séquences communes inversées (CIS pour *common inverted subsequences*) :

$$p_1 \prec_3 p_2 \Leftrightarrow \Delta i > 0, \Delta j < 0$$

- Forêts communes inversées (CIF pour *common inverted forests*) :

$$p_1 \prec_4 p_2 \Leftrightarrow \Delta i \geq 0, \Delta j \leq 0$$

Soit $\mathcal{C}$ une famille de sous-ensembles d'un ensemble P. Une sous-famille $\mathcal{C}' \subseteq \mathcal{C}$ est appelée un *recouvrement* de P si chaque $p \in P$ est contenu dans au moins un des sous-ensembles $C \in \mathcal{C}'$. Le nombre d'éléments dans $\mathcal{C}'$ est appelé le *cardinal* du recouvrement $\mathcal{C}'$ et un recouvrement de cardinal minimal est appelé un *recouvrement minimal* de P par $\mathcal{C}$. Le théorème suivant a été prouvé par Dilworth, 1950 [86].

Théorème 6.4 *Soit P un ensemble partiellement ordonné. Le cardinal d'un recouvrement minimal de P par des chaînes est égal au cardinal d'une antichaîne maximale dans P.*

Lemme 6.3 *Soient $\prec$ et $\prec^*$ des ordres partiels conjugués dans P. Alors la longueur d'une des plus longues $\prec$-séquences dans P est égal au cardinal d'un recouvrement minimal de P par des $\prec^*$-séquences.*

Preuve D'après le théorème de Dilworth, la longueur d'une des plus longues antichaînes dans $\prec^*$ est égale au cardinal d'un recouvrement minimal de P par des $\prec^*$-chaînes. Comme $\prec$ et $\prec^*$ sont conjugués, toute antichaîne dans $\prec^*$ est une chaîne dans $\prec$ et toute chaîne dans $\prec$ est une antichaîne dans $\prec^*$. Par conséquent, la longueur d'une des plus longues $\prec$-séquences dans P est égale au cardinal d'un recouvrement minimal de P par des $\prec^*$-séquences. ■

Comme les CS et CIF représentent des ordres partiels conjugués, le lemme 6.3 implique le théorème suivant (Pevzner et Waterman, 1993 [273]) :

Théorème 6.5 *La longueur des plus longues CS est égale au cardinal d'un recouvrement minimal par les CIF.*

Considérons la relation binaire sur P définie par :

$$p_1 \sqsubset p_2 \Longleftrightarrow p_1 \prec p_2 \text{ ou } p_1 \prec^* p_2.$$

Lemme 6.4 *La relation $\sqsubset$ est un ordre total sur P.*

Preuve Les caractères réflexif et antisymétrique de la relation sont immédiats. Seule la transitivité est délicate. Si $p_1 \sqsubset p_2$ et $p_2 \sqsubset p_3$, alors l'une des conditions suivantes est vérifiée :

(i) $p_1 \prec p_2$ et $p_2 \prec p_3$,

(ii) $p_1 \prec p_2$ et $p_2 \prec^* p_3$,

(iii) $p_1 \prec^* p_2$ et $p_2 \prec p_3$,

(iv) $p_1 \prec^* p_2$ et $p_2 \prec^* p_3$.

Dans le cas (i), la transitivité de $\prec$ implique $p_1 \prec p_3$ et, par conséquent, $p_1 \sqsubset p_3$. Dans le cas (ii), $p_1 \prec p_2$ et $p_2 \prec^* p_3$ n'impliquent ni $p_3 \prec p_1$, ni $p_3 \prec^* p_1$. (Dans le premier cas, $p_3 \prec p_1$ et $p_1 \prec p_2$ impliquent $p_3 \prec p_2$, ce qui contredit $p_2 \prec^* p_3$. Dans le second cas, $p_2 \prec^* p_3$ et $p_3 \prec^* p_1$ impliquent $p_2 \prec^* p_1$, contredisant $p_1 \prec p_2$). On a donc $p_1 \prec p_3$ ou $p_1 \prec^* p_3$, ce qui implique $p_1 \sqsubset p_3$. Notons que les cas (iii) et (iv) sont symétriques aux cas (ii) et (i) respectivement, nous avons

donc montré que la relation $\sqsubset$ est transitive. Le lemme se déduit de l'observation selon laquelle, pour toute paire (p_1, p_2), on a soit $p_1 \sqsubset p_2$, soit $p_2 \sqsubset p_1$. ■

6.10 Approche primale-duale de la comparaison de séquences

Le théorème 6.5 révèle une relation entre la LCS et le problème du recouvrement minimal et donne l'idée d'utiliser des recouvrements minimaux pour trouver la LCS. Nous décrivons à présent un algorithme de programmation non dynamique qui donne simultanément une solution au problème de l'alignement généralisé et à celui du recouvrement minimal (Pevzner et Waterman, 1993 [273]).

Soit $\mathcal{P} = p_1 p_2 \dots p_l$ un ordre arbitraire des éléments de P et soit $\mathcal{P}_i = p_1 p_2 \dots p_i$. Soit $\mathcal{C}_i = \{C_1, C_2, \dots, C_j\}$ un recouvrement de $\mathcal{P}_i$ par des $\prec^*$-séquences et soient $p_1^{max}, p_2^{max}, \dots, p_j^{max}$ les éléments $\prec^*$-maximaux dans C_1, $C_2, \dots, C_j$, respectivement. Soit k l'indice minimal $(1 \le k \le j)$ remplissant la condition suivante :

$$p_k^{max} \prec^* p_{i+1}. \tag{6.1}$$

Si la condition (6.1) n'est vérifiée pour aucun k $(1 \le k \le j)$, on pose $k = j+1$. Pour plus de commodité, on définit C_{j+1} comme étant l'ensemble vide. L'algorithme RECOUVREMENT construit un recouvrement $\mathcal{C}_{i+1}$ à partir de $\mathcal{C}_i$ en adjoignant p_{i+1} à C_k. Si k est strictement inférieur à $j+1$, RECOUVREMENT agrandit C_k :

$$\mathcal{C}_{i+1} = \{C_1, C_2, \dots, C_{k-1}, C_k \cup \{p_{i+1}\}, C_{k+1}, \dots, C_j\}.$$

Si k est égal à $j+1$, RECOUVREMENT ajoute $\{p_{i+1}\}$ comme une nouvelle $\prec^*$-séquence au recouvrement $\mathcal{C}_{i+1}$:

$$\mathcal{C}_{i+1} = \{C_1, C_2, \dots, C_j, C_{j+1} = \{p_{i+1}\}\ \}.$$

L'algorithme conserve également un *retour arrière* :

$$b(p_{i+1}) = \begin{cases} p_{k-1}^{max}, & \text{si } k > 1 \\ \emptyset, & \text{sinon.} \end{cases}$$

En commençant avec un recouvrement vide $\mathcal{C}_0$, RECOUVREMENT construit un recouvrement $\mathcal{C}_\ell$ de P après ℓ itérations, dont le cardinal dépend de l'ordonnancement de P. Le cardinal du recouvrement $\mathcal{C}_\ell$ est une borne supérieure pour la longueur de la plus longue $\prec$-séquence. Le théorème suivant montre que, si $\mathcal{P}$ est l'ordonnancement de P dans $\sqsubset$, alors RECOUVREMENT est un algorithme primal-dual permettant de résoudre simultanément le problème de la plus longue $\prec$-séquence et celui du recouvrement minimal par des $\prec^*$-séquences.

Théorème 6.6 *Si $\mathcal{P} = p_1 p_2 \dots p_\ell$ est l'ordonnancement de P dans $\sqsubset$, alors RECOUVREMENT construit un recouvrement minimal $\mathcal{C}_\ell = \{C_1, C_2, \dots, C_t\}$ de P par des $\prec^*$-séquences. Le retour arrière $b(p)$ définit l'une des plus longues $\prec$-séquences de longueur t pour chaque $p \in C_t$.*

Preuve On montre par récurrence que, pour tout i ($1 \leq i \leq \ell$), le recouvrement $\mathcal{C}_i = \{C_1, C_2, \dots, C_j\}$ vérifie la condition

$$\forall k > 1, \forall p \in C_k : \quad b(p) \prec p. \tag{6.2}$$

Cette condition est trivialement vraie pour $\mathcal{C}_1$. Supposons alors qu'elle soit vraie pour $\mathcal{C}_i$ et montrons qu'il en est de même pour $\mathcal{C}_{i+1}$. On considère deux cas :

– *Cas 1.* $k < j + 1$ (condition (6.1)). Dans ce cas, on a $b(p_{i+1}) = p_{k-1}^{max}$. Comme $\mathcal{P}$ est le $\sqsubset$-ordonnancement, on en déduit $p_{k-1}^{max} \sqsubset p_{i+1}$; par conséquent, on a soit $p_{k-1}^{max} \prec p_{i+1}$, soit $p_{k-1}^{max} \prec^* p_{i+1}$. Or k est l'indice minimal remplissant la condition $p_k^{max} \prec^* p_{i+1}$, d'où $p_{k-1}^{max} \prec p_{i+1}$ et la condition (6.2) est vraie pour $\mathcal{C}_{i+1}$.

– *Cas 2.* $k = j + 1$. Dans ce cas, on a $b(p_{i+1}) = p_j^{max}$. Puisque $\mathcal{P}$ est le $\sqsubset$-ordonnancement, on a soit $p_j^{max} \prec p_{i+1}$, soit $p_j^{max} \prec^* p_{i+1}$. Comme k est égal à $j + 1$, la condition (6.1) n'est vérifiée pour aucun $k \leq j$. Par conséquent, on obtient $p_j^{max} \prec p_{i+1}$ et la condition (6.2) est vraie pour $\mathcal{C}_{i+1}$.

Il est évident que chaque recouvrement $\mathcal{C}_\ell = \{C_1, C_2, \dots, C_t\}$ qui vérifie la condition (6.2) détermine (au moyen du retour arrière) une $\prec$-séquence de longueur t pour tout $p \in C_t$. D'après le lemme 6.3, une telle séquence est l'une des plus longues pour l'ordre $\prec$ et $\mathcal{C}_\ell$ est un recouvrement minimal de P par des $\prec^*$-séquences. ■

Pour la LCS entre des séquences à n lettres, l'algorithme RECOUVREMENT peut être implémenté en un temps $O(nL)$, où L est la longueur d'une des plus longues sous-séquences communes, ou en un temps $O((\ell + n) \log n)$, où ℓ est le nombre total d'égalités entre deux séquences. Hirschberg, 1977 [164] et Hunt et Szymanski, 1977 [174] ont suggéré des améliorations à l'algorithme de programmation dynamique classique permettant de trouver la LCS. En fait, l'algorithme de Hirschberg et celui de Hunt-Szymanski peuvent être vus comme des implémentations de l'algorithme RECOUVREMENT avec des structures de données différentes. Les $\prec^*$-chaînes dans RECOUVREMENT correspondent aux *k-candidats* dans l'algorithme de Hirschberg. Les éléments maximaux des $\prec^*$-chaînes dans RECOUVREMENT correspondent aux *égalités dominantes* dans la version améliorée d'Apostolico, 1986 [9] de l'algorithme de Hunt-Szymanski.

6.11 Alignement de séquences et programmation en nombres entiers

La dualité pour la LCS est étroitement liée à une nouvelle approche *polyédrique* de la comparaison de séquences suggérée par Reinert *et al.*, 1997 [283]. Ils expriment le problème de l'alignement sous la forme d'un programme linéaire en nombres entiers et rapportent que cette approche de l'alignement multiple peut résoudre des exemples que la programmation dynamique ne permettait pas de résoudre. Ci-dessous est décrite la relation entre le problème de la LCS et la programmation en nombres entiers.

Soit P un ensemble muni d'un ordre partiel $\prec$ et soit $\prec^*$ un ordre partiel conjugué. Soit x_e le poids associé à $e \in P$. Dans le cas de la LCS des séquences $v_1 \dots v_n$ et $w_1 \dots w_m$, P est l'ensemble des paires de positions $e = (i, j)$ telles que $v_i = w_j$ et $x_e = 1$ si et seulement si les positions i et j sont appariées dans la LCS. Le problème de la LCS peut être formulé sous la forme du problème de programmation en nombres entiers suivant :

$\sum_{x_e \in \alpha} x_e \leq 1$ pour toute antichaîne (maximale) α dans P
$\sum_{e \in P} x_e \to \max$

Soit y_α une variable associée à une antichaîne maximale α dans P. Le programme *dual* du problème ci-dessus est le suivant :

$\sum_{e \in \alpha} y_\alpha \geq 1$ pour tout $e \in P$
$\sum_{\alpha} y_\alpha \to \min$

Comme les antichaînes dans $\prec$ sont des chaînes dans $\prec^*$, le programme ci-dessus est le *problème du recouvrement de chemin minimal* dans un graphe orienté acyclique représentant l'ordre partiel conjugué $\prec^*$, problème connu pour avoir une solution entière (Cormen *et al.*, 1989 [75]).

6.12 Appariement de chaînes approximatif

L'appariement de chaînes approximatif avec k mésappariements comprend une chaîne $t_1 \dots t_n$, que l'on appelle le *texte*, une chaîne plus courte $q_1 \dots q_p$, appelée la *requête* et des entiers k et m. Le *problème de l'appariement requête* est de trouver toutes les m-sous-chaînes de la requête $q_i \dots q_{i+m-1}$ et du texte $t_j \dots t_{j+m-1}$ qui s'apparient avec au plus k mésappariements. Dans le cas $p = m$, le problème de l'*appariement requête* donne le *problème de l'appariement de chaînes approximatif avec k mésappariements.*

Le problème de l'appariement de chaînes approximatif avec k mésappariements a été étudié intensivement en informatique. Pour $k = 0$, il se ramène à l'appariement de chaînes classique, résoluble en un temps $O(n)$ (Knuth *et al.*, 1977 [203] et Boyer et Moore, 1977 [45]). Pour $k > 0$, l'algorithme naïf de force brute pour l'appariement de chaînes approximatif se fait en un temps $O(nm)$. Des algorithmes linéaires pour l'appariement de chaînes approximatif

ont été décrits par Ivanov, 1984 [177] et Landau et Vishkin, 1985 [213]. Pour un alphabet de taille fixée, la pire complexité temporelle de ces algorithmes est $O(kn)$.

Bien que ces algorithmes donnent la meilleure performance dans le pire des cas, ils sont loin d'être les meilleurs dans la pratique (Grossi et Luccio, 1989 [140]). Par conséquent, plusieurs approches fondées sur le filtrage ont attiré l'attention sur la complexité temporelle moyenne, en contraste avec celle du pire des cas (Baeza-Yates et Gonnet, 1989 [16], Grossi et Luccio, 1989 [140], Tarhio et Ukkonen, 1990 [334], Baeza-Yates et Perleberg, 1992 [17] et Wu et Manber, 1992 [371]).

L'utilisation d'algorithmes de filtrage pour l'appariement de chaînes approximatif implique un procédé à deux phases. La première d'entre elles présélectionne un ensemble de positions dans le texte qui sont *potentiellement* semblables à la requête. La seconde phase vérifie chaque position potentielle, rejetant les appariements ayant plus de k mésappariements. On note p le nombre d'appariements potentiels trouvés lors de la première phase de l'algorithme. La présélection est habituellement faite en un temps $\alpha n + O(p)$, où α est une petite constante. Si le nombre d'appariements potentiels est petit et si la vérification d'un appariement potentiel n'est pas trop longue, cette méthode conduit à une accélération significative.

L'idée d'utiliser le filtrage pour résoudre le problème de l'appariement de chaînes a d'abord été décrite par Karp et Rabin, 1987 [189] dans le cas $k = 0$. Pour $k > 0$, Owolabi et McGregor, 1988 [258] ont utilisé l'idée de *l-uple filtrage* fondée sur la simple observation selon laquelle, si une requête s'apparie approximativement à un texte, alors ils ont au moins un l-uplet commun, pour un l suffisamment grand. Tous les l-uplets communs à la requête et au texte peuvent être trouvés facilement par hachage. Si le nombre de l-uplets communs est relativement petit, ils peuvent être vérifiés et tous les appariements *réels* avec k mésappariements peuvent être repérés rapidement.

Le l-uple filtrage est fondé sur la simple observation suivante :

Lemme 6.5 *Si les chaînes* $x_1 \ldots x_m$ *et* $y_1 \ldots y_m$ *s'apparient avec au plus* k *mésappariements, alors elles ont un l-uplet commun, pour* $l = \lfloor \frac{m}{k+1} \rfloor$*, c'est-à-dire* $x_i \ldots x_{i+l-1} = y_j \ldots y_{j+l-1}$ *pour un certain couple* (i, j) *vérifiant* $1 \leq i, j \leq m - l + 1$.

Ce lemme motive un algorithme de *l-uple filtrage* pour l'appariement requête avec k mésappariements :

Algorithme FILTRAGE Détection de tous les m-appariements entre une requête et un texte avec un maximum de k mésappariements.

– *Détection des appariements potentiels.* Trouver tous les appariements de l-uplets dans la requête et le texte pour $l = \lfloor \frac{m}{k+1} \rfloor$.

– *Vérification des appariements potentiels.* Vérifier chaque appariement potentiel en l'étendant à gauche et à droite jusqu'à ce que (i) les premiers $k+1$ mésappariements soient trouvés ou (ii) le début ou la fin de la requête ou du texte soit trouvé.

Le lemme 6.5 garantit que FILTRAGE trouve *tous* les appariements de longueur m ayant au maximum k mésappariements. La détection d'un appariement potentiel dans FILTRAGE peut être implémentée par hachage. La complexité temporelle de FILTRAGE est en $\alpha n + O(pm)$, où p est le nombre d'appariements potentiels détectés durant la première phase de l'algorithme et α une petite constante. Pour un texte de Bernoulli avec A lettres équiprobables, le nombre attendu d'appariements potentiels est d'environ $E(p) = \frac{nq}{A^l}$, ce qui donne un algorithme rapide pour de grandes valeurs de A et de l.

6.13 Recherche d'une séquence dans une base de données

Une *matrice de points* de deux séquences V et W est simplement une matrice dont chaque terme vaut 0 ou 1 : si elle présente un 1 en position (i, j), cela indique que les l-uplets commençant à la i-ième position de V et à la j-ième position de W coïncident. FASTA (Lipman et Pearson, 1985 [225]), un outil de recherche populaire dans les bases de données protéiques, utilise habituellement le l-uple filtrage avec $l = 2$ (dans l'alphabet des acides aminés). Les positions des l-uplets présents dans les deux chaînes forment une représentation d'une matrice de points (implicite) des similitudes entre les chaînes. FASTA assemble ensuite les 1 situés sur une même diagonale de cette matrice de points et tente de regrouper les diagonales voisines.

Utiliser des l-uplets communs pour trouver des similitudes comporte quelques inconvénients. BLAST (Altschul *et al.*, 1990 [5]), l'outil qui domine dans la recherche sur bases de données en biologie moléculaire, utilise des matrices de substitution afin d'améliorer la construction de matrices de points (implicites) pour l'analyse ultérieure des diagonales. Il tente essentiellement de perfectionner le filtrage de FASTA en introduisant des règles plus strictes, afin que les appariements potentiels localisés soient en plus petit nombre et de meilleure qualité. Une autre caractéristique de BLAST est l'utilisation des statistiques d'Altschul-Dembo-Karlin (Karlin et Altschul, 1990 [186] et Dembo et Karlin, 1991 [85]) pour les estimations. Pour toute paire de l-uplets $x_1 \ldots x_l$ et $y_1 \ldots y_l$, BLAST définit le *score d'un segment* comme étant égal à $\sum_{i=1}^{l} \delta(x_i, y_i)$, où $\delta(x, y)$ est le score de similitude entre les acides aminés x et y. Une « *paire de segments maximale* » (MSP pour *Maximal Segment Pair*) est une paire de l-uplets ayant le score maximal parmi toutes les paires de segments dans deux séquences. Un biologiste moléculaire peut être intéressé par tous les segments observés, pas seulement par les paires ayant le score le plus élevé. Une paire de

segments est dite *localement maximale* si son score ne peut être amélioré, que ce soit en étendant ou en racourcissant les deux segments.

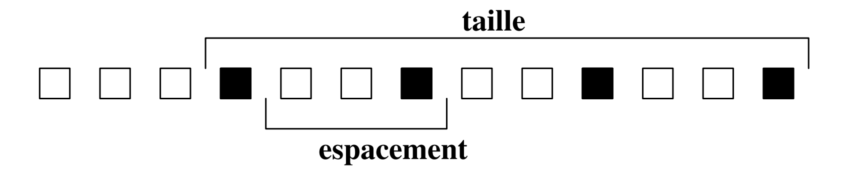
BLAST tente de trouver les paires de segments localement maximales dans la séquence requête et la base de données avec des scores dépassant un certain seuil. Le choix du seuil est guidé par les statistiques d'Altschul-Dembo-Karlin, qui permettent d'identifier la plus faible valeur de score d'un segment qui a peu de chance de se produire. BLAST donne des séquences qui ont un score d'un segment supérieur au seuil ou des séquences dont le score ne dépasse pas le seuil mais qui ont plusieurs paires de segments statistiquement significatives une fois combinées.

BLAST abandonne l'idée de l-uple filtrage et utilise une stratégie différente pour trouver des appariements potentiels. Il trouve tous les l-uplets ayant des scores supérieurs à un certain seuil avec un l-uplet dans la requête. Ceci peut être fait soit directement — en trouvant toutes les occurrences approximatives des sous-chaînes de la requête dans la base de données — soit de façon plus compliquée. Par exemple, si le seuil est suffisamment élevé, alors l'ensemble de ces chaînes n'est pas trop grand et on peut en chercher des occurrences exactes dans la base de données. Ceci est un problème combinatoire d'appariement très étudié et l'algorithme rapide d'Aho et Corasick, 1975 [1] localise les occurrences de *toutes* ces chaînes dans la base de données. Après avoir situé les appariements potentiels, BLAST tente de les étendre pour voir si le score obtenu est supérieur au seuil. Altschul *et al.*, 1997 [7] ont encore amélioré BLAST en autorisant des insertions et des délétions et en combinant des appariements sur les mêmes diagonales et celles qui sont proches.

6.14 Filtrage multiple

Pour des textes de Bernoulli avec A lettres équiprobables, on définit l'*efficacité de filtrage* d'un algorithme de filtrage comme étant le quotient $\frac{E(r)}{E(p)}$ du nombre attendu d'appariements avec k mésappariements $E(r)$ par le nombre attendu d'appariements potentiels $E(p)$. Par exemple, pour $k = 1$, l'efficacité du l-uple filtrage, qui vaut environ $\frac{A-1}{A^{\lceil \frac{m}{2} \rceil}}$, décroît rapidement lorsque m et A augmentent. Cette observation montre l'intérêt de trouver une méthode de filtrage ayant une efficacité accrue. Plus le taux d'efficacité est grand, plus la complexité temporelle de la phase de vérification est courte pour l'algorithme de filtrage.

Pevzner et Waterman, 1995 [274] ont décrit un algorithme qui permet la réduction exponentielle du nombre d'appariements potentiels contre une augmentation linéaire du temps de filtrage. Ceci réduit de façon significative le temps de la phase de vérification de l'algorithme FILTRAGE qui croît linéairement lors de la phase de détection. En tenant compte du fait que la vérification est souvent plus coûteuse en temps que la détection, la technique fournit une possibilité de choix optimal des paramètres de filtrage.



4-uplet avec trous d'espacement 3 et de taille 10 debutant en position 4

Figure 6.7 – Un 4-uplet avec trous.

Un ensemble de positions $i, i+t, i+2t, \ldots, i+jt, \ldots, i+(l-1)t$ est appelé un *l-uplet avec trous* d'*espacement* t et de *taille* $1+t(l-1)$ (voir figure 6.7). Des l-uplets continus peuvent être vus comme des l-uplets avec trous d'espacement 1 et de taille l. Si un l-uplet commun à un modèle et à un texte commence en la position i du modèle et en la position j de la requête, on dit que (i, j) sont les *coordonnées* du l-uplet. On définit la *distance* $d(v_1, v_2)$ entre des l-uplets de coordonnées (i_1, j_1) et (i_2, j_2) de la façon suivante :

$$d(v_1, v_2) = \begin{cases} i_1 - i_2, & \text{si } i_1 - i_2 = j_1 - j_2 \\ \infty, & \text{sinon.} \end{cases}$$

Le filtrage multiple est fondé sur l'observation suivante :

Lemme 6.6 *Soient $x_1 \ldots x_m$ et $y_1 \ldots y_m$ deux chaînes qui s'apparient avec au plus k mésappariements et soit $l = \lfloor \frac{m}{k+1} \rfloor$. Alors ces chaînes ont en commun un l-uplet continu et un l-uplet avec trous d'espacement $k+1$, situés à distance d l'un de l'autre et vérifiant la condition $-k \leq d \leq m-l$.*

Ce lemme conduit à l'algorithme de *double filtrage* pour l'appariement de requête avec k mésappariements suivant :

Algorithme DOUBLE-FILTRAGE Détection de tous les m-appariements entre une requête et un texte avec au plus k mésappariements.

– *Détection d'appariements potentiels.* Trouver tous les appariements de l-uplets continus entre la requête et le texte qui sont situés à distance $-k \leq d \leq m-l$ d'un appariement de l-uplets avec trous d'espacement $k+1$.

– *Vérification des appariements potentiels.* Vérifier chaque appariement potentiel en l'étendant à gauche et à droite jusqu'à ce que soit (i) les $k+1$ premiers mésappariements soient trouvés, soit (ii) le début ou la fin de la requête ou du texte soit trouvé.

Le lemme 6.6 garantit que DOUBLE-FILTRAGE trouve *tous* les appariements avec k mésappariements. L'efficacité du double filtrage est environ

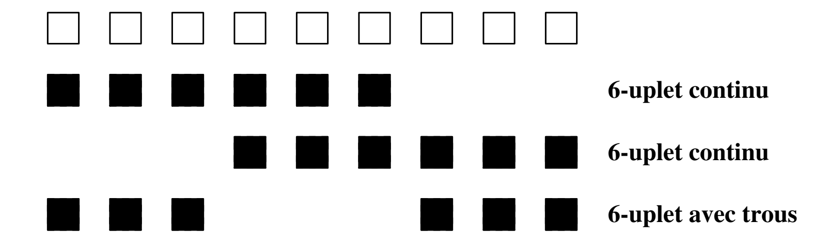


Figure 6.8 – Tout 9-appariement avec une erreur contient soit un 6-appariement continu, soit un 6-appariement avec trous.

$\frac{A^{l-\delta}}{m(1-\frac{1}{A})}$ fois plus grande que l'efficacité du l-uple filtrage pour un grand nombre de paramètres (ici, $\delta = \lceil \frac{l}{k+1} \rceil$).

L'appariement de modèles approximatif de 9-uplets avec une erreur peut se restreindre à l'appariement exact de 4-uplets. On l'appelle une réduction d'un appariement de modèle (9,1) à un appariement de modèle (4,0). Peut-on restreindre un appariement de modèle (9,1) à un appariement de modèle (6,0), augmentant ainsi l'efficacité du filtrage ? La réponse est affirmative, si l'on considère des 6-uplets avec trous comme indiqué dans la figure 6.8. On peut également se demander s'il est possible d'obtenir une accélération par la restriction de l'appariement de modèle (m, k) à l'appariement (m', k'), pour $0 < k' < k$. L'exploration de ce problème a abouti au développement d'algorithmes d'appariement de modèles sublinéaires (Chang et Lawler, 1994 [62] et Myers, 1994 [245]).

6.15 Quelques autres problèmes et approches

6.15.1 Alignement de séquences paramétrique

L'alignement de séquences est sensible au choix des pénalités d'insertion/délétion et de substitution ; un choix incorrect de ces paramètres peut mener à des alignements biologiquement incorrects. Comprendre l'influence des paramètres sur l'alignement obtenu et choisir les alignements appropriés sont deux choses très importantes pour les applications biologiques. Dans le modèle le plus simple, lorsque le score d'alignement est défini par la formule « nombre d'appariements – $\mu\times$ nombre de mésappariements – $\sigma\times$ nombres d'indels », différentes valeurs de (μ, σ) correspondent à différents alignements optimaux. Cependant, certaines régions dans l'espace des (μ, σ) correspondent au même alignement optimal. En fait, l'espace des (μ, σ) peut être décomposé en polygones convexes de telle sorte que tout couple de points d'un même polygone correspondent au même alignement optimal (Fitch et Smith, 1983 [107]). Waterman *et al.*, 1992 [360], Gusfield *et al.*, 1994 [146] et Zimmer et Lengauer, 1997 [379] ont décrit des algorithmes efficaces permettant de calculer une décomposition polygonale de l'espace des paramètres.

6.15.2 Statistiques d'alignement et transition de phase

Arratia et Waterman, 1989 [13] ont étudié les propriétés statistiques du score d'alignement local de deux séquences aléatoires. Il s'est avéré que les statistiques d'alignement dépendent fortement du choix des pénalités d'indel et d'appariement. Le score d'alignement local croît d'abord logarithmiquement, puis linéairement avec la longueur des séquences, en fonction des pénalités de trous. On désigne ces deux régions de croissance comme étant la « région log » et la « région linéaire » et la courbe située entre celles-ci est appelée la courbe de *transition de phase*.

Soient $v_1 \dots v_n$ et $w_1 \dots w_n$ deux chaînes i.i.d. Soient $S_n = S_n(\mu, \delta)$ et $H_n = H_n(\mu, \delta)$ deux variables aléatoires correspondant au score (nombre d'appariements − $\mu\times$ nombre de mésappariements − $\sigma\times$ nombres d'indels) de l'alignement global et de l'alignement local de ces chaînes. Arratia et Waterman, 1994 [14] ont montré que

$$a = a(\mu, \delta) = \lim_{n\to\infty} \frac{S_n}{n}$$

existe en probabilité. De plus, $\{a = 0\} = \{(\mu, \delta) : a(\mu, \delta) = 0\}$ est une courbe de transition de phase qui sépare $[0, \infty]^2$ en deux composantes : $\{a < 0\} = \{(\mu, \delta) : a(\mu, \delta) < 0\}$ et $\{a > 0\} = \{(\mu, \delta) : a(\mu, \delta) > 0\}$.

Dans le cas $(\mu, \delta) \in \{a > 0\}$, Arratia et Waterman, 1994 [14] ont montré l'égalité : $\lim_{n\to\infty} \frac{H_n(\mu,\delta)}{n} = a(\mu, \delta)$. Dans le cas $(\mu, \delta) \in \{a < 0\}$, ils ont introduit une constante $b = b(\mu, \delta)$ de façon à avoir

$$\lim_{n\to\infty} \mathbf{P}\{(1-\epsilon)b < \frac{H_n(\mu,\delta)}{\log(n)} < (2+\epsilon)b\} = 1.$$

Le problème consistant à calculer $a(\mu, \delta)$ pour $(\mu, \delta) \in \{a > 0\}$ est difficile. En particulier, calculer $a(0, 0)$ résoudrait la conjecture de Steele-Arratia. On peut consulter Vingron et Waterman, 1994 [346] et Bundschuh et Hwa, 1999 [52] pour des estimations de l'importance de l'alignement et le choix de paramètres dans la « région log » et la « région linéaire » ; on peut également lire Waterman et Vingron, 1994 [365] pour un algorithme rapide qui calcule la probabilité pour qu'un score d'alignement local soit uniquement le résultat du hasard.

6.15.3 Alignement de séquences sous-optimal

Parfois, l'alignement optimal n'est pas celui qui est biologiquement correct et l'on imagine des méthodes pour créer un ensemble d'alignements Δ-sous-optimaux qui diffèrent de l'optimal d'au plus Δ (Waterman, 1983 [355]). Le problème équivaut à trouver des chemins sous-optimaux dans le graphe d'édition (Chao, 1994 [63] et Naor et Brutlag, 1994 [250]).

6.15.4 Alignement avec duplications en tandem

La plupart des algorithmes d'alignement ne considèrent que les insertions, les délétions et les substitutions. Il peut cependant se produire d'autres événements mutationnels. La *duplication en tandem* en est un : dans ce cas, une section d'ADN est dupliquée pour produire une ou plusieurs nouvelles copies, chacune d'elles étant contiguë à la précédente. Cette mutation est plutôt commune ; on estime qu'elle occupe environ 10% du génome humain. Les répétitions en tandem ont été impliquées dans un certain nombre de maladies humaines héréditaires, y compris la maladie de Huntington. Benson, 1997 [30] a proposé un algorithme efficace pour l'alignement de séquences avec des duplications en tandem. Des algorithmes permettant de *détecter* des répétitions en tandem ont été suggérés par Landau et Schmidt, 1993 [212], Milosavljevic et Jurka, 1993 [237] et Benson, 1998 [31].

6.15.5 Résultats de la recherche dans des bases de données passées au crible

Dans la recherche dans des bases de données, les appariements à des régions biologiquement importantes sont souvent cachés par d'autres appariements. Un grand nombre d'appariements dans une région de la séquence peut cacher des appariements de score plus faible mais importants qui se produisent ailleurs. Comme les programmes de recherche dans des bases de données donnent souvent un nombre fixé d'appariements supérieurs et tronquent la sortie, des règles sont nécessaires pour sélectionner un sous-ensemble d'appariements qui révèle tous les résultats importants. Le problème est modélisé par une liste d'intervalles (régions d'alignement) avec des scores d'alignement associés. Si l'intervalle I est contenu dans l'intervalle J avec un score supérieur, alors I est *dominé* par J. Le *problème du passage au crible* est d'identifier et de mettre de côté les intervalles qui sont dominés par un nombre fixé d'autres intervalles. Berman *et al.*, 1999 [34] ont implémenté une version de BLAST qui résout le problème du passage au crible en un temps $O(n \log n)$, où n désigne le nombre d'intervalles.

6.15.6 Distance statistique entre des textes

Soit $\mathcal{X}$ un ensemble de chaînes — par exemple, l'ensemble constitué de tous les l-uplets (Blaisdell, 1988 [36]) ou des l-uplets avec trous (Mironov et Alexandrov, 1988 [239]), pour l petit. Étant donnés une chaîne $x \in \mathcal{X}$ et un texte T, on définit $x(T)$ comme étant le nombre (ou la fréquence) d'occurrences de x dans T. Blaisdell, 1988 [36] et Mironov et Alexandrov, 1988 [239] ont défini la *distance statistique* entre des textes V et W :

$$d(V, W) = \sqrt{\sum_{x \in \mathcal{X}} (x(V) - x(W))^2},$$

bien avant que les moteurs de recherche sur Internet commencent à utiliser des mesures voisines pour trouver des pages similaires sur le Web. Pevzner, 1992 [266] a étudié l'efficacité de la distance statistique pour trouver des similitudes. L'avantage de la méthode de la distance statistique comparé à BLAST est la vitesse : la distance statistique peut être calculée très rapidement avec le prétraitement des bases de données. L'inconvénient est que la distance statistique peut manquer des faibles similitudes qui ne préservent pas les l-uplets communs. En conséquence, l'application majeure de tels algorithmes se situe dans des comparaisons base de données contre base de données, comme le regroupement d'EST. Pour obtenir la très grande vitesse nécessaire pour les grandes bases de données d'EST, l'approche de la distance statistique a été récemment implémentée avec des tables de suffixes (Burkhardt *et al.*, 1999 [55]).

6.15.7 Repliement de l'ARN

Les ARN adoptent des structures tridimensionnelles sophistiquées qui sont importantes dans la reconnaissance du signal et la régulation génétique. Les paires de positions dans l'ARN avec des bases de Watson-Crick complémentaires peuvent former des liaisons. Des liaisons (i, j) et (i', j') se chevauchent si l'on a $i < i' < j < j'$; elles sont non chevauchantes dans le cas contraire. Dans une formulation très naïve du problème du repliement de l'ARN, on essaie de trouver un ensemble maximal de liaisons non chevauchantes. Le problème peut être résolu par la programmation dynamique (Nussinov *et al.*, 1978 [253] et Waterman, 1978 [354]). Dans un modèle plus précis, on tente de trouver un repliement de l'ARN d'énergie minimale (Zuker et Sankoff, 1984 [381], Waterman et Smith, 1986 [363] et Zuker, 1989 [380]). Cependant, ces algorithmes ne sont pas très sûrs. Une approche plus prometteuse consiste à déduire un repliement de l'ARN à partir d'un alignement multiple de molécules d'ARN voisines. Eddy et Durbin, 1994 [95] ont étudié un problème d'alignement multiple de l'ARN qui prend en compte l'information du repliement.

Chapitre 7

Alignement multiple

7.1 Introduction

Le but de la comparaison de séquences protéiques est de découvrir des similitudes « biologiques » (i.e. structurelles ou fonctionnelles) parmi les protéines. Des protéines biologiquement similaires peuvent ne pas exhiber une forte similitude de séquences et l'on aimerait reconnaître la ressemblance structurelle/fonctionnelle, même lorsque les séquences sont très différentes. Si la similitude de séquences est faible, l'alignement par paires peut ne pas identifier des séquences apparentées biologiquement, car de faibles similitudes au niveau des paires peuvent faire échouer les tests statistiques. La comparaison simultanée de nombreuses séquences permet souvent de trouver des similitudes invisibles dans la comparaison de séquences par paires. Pour citer Hubbard *et al.*, 1996 [170] « l'alignement par paires chuchote... l'alignement multiple crie ».

La programmation dynamique directe résout le problème de l'alignement multiple pour k séquences de longueur n. Comme la complexité temporelle de cette approche est en $O((2n)^k)$, on a imaginé à partir de l'algorithme de base un certain nombre de variantes et quelques accélérations (Sankoff, 1975 [298], Sankoff, 1985 [299] et Waterman *et al.*, 1976 [364]). Cependant, les algorithmes d'alignement multiple ont une trop grande complexité pour pouvoir être utilisés pour de grandes valeurs de k (Wang et Jiang, 1994 [351]) et de nombreuses heuristiques ont été proposées pour l'alignement multiple sous-optimal.

Une heuristique naturelle consiste à calculer les alignements optimaux des C_k^2 paires obtenues à partir des k séquences et à les combiner de sorte que les alignements induits soient proches des optimaux. Malheureusement, il n'est pas toujours possible de combiner des alignements par paires en des alignements multiples, car certains peuvent être incompatibles. Ainsi, de nombreux algorithmes d'alignement multiple tentent de combiner un certain sous-ensemble compatible d'alignements par paires optimaux en un alignement multiple. Ceci peut être réalisé pour de petits sous-ensembles de tous les C_k^2 alignements par

paires. Le problème est de décider quel sous-ensemble d'alignements par paires choisir pour cette procédure.

L'approche la plus simple utilise l'alignement par paires pour ajouter itérativement une séquence à un alignement multiple croissant. Feng et Doolittle, 1987 [100] utilisent la paire de séquences de plus grande similitude et les fusionne en une nouvelle séquence suivant le principe « une brêche un jour, une brêche toujours ». Ainsi, l'alignement multiple de k séquences se ramène à l'alignement multiple de $k-1$ séquences. De nombreux autres algorithmes itératifs d'alignement multiple utilisent des stratégies semblables (Barton et Sternberg, 1987 [27], Taylor, 1987 [336], Bains, 1986 [22] et Higgins *et al.*, 1996 [162]).

Bien que l'algorithme de Feng et Doolittle, 1987 [100] fonctionne bien pour des séquences proches, il n'y a pas de garantie de performance pour cette méthode. Le premier algorithme d'approximation à performance garantie pour l'alignement multiple, avec un taux d'approximation de $2 - \frac{2}{k}$, a été proposé par Gusfield, 1993 [144]. L'idée de l'algorithme repose sur la notion d'alignements compatibles et utilise le principe « une brêche un jour, une brêche toujours ».

Feng et Doolittle, 1987 [100] et Gusfield, 1993 [144] utilisent les alignements optimaux par paires (niveau 2) comme des blocs de construction pour des alignements multiples (niveau k). Une extension naturelle de cette approche consiste à utiliser les alignements optimaux de niveau 3 (ou l) en tant que blocs de construction pour des alignements de niveau k. Cependant, cette approche pose des problèmes d'ordre combinatoire, car la façon de définir des alignements de niveau l compatibles et la manière de les combiner ne sont pas évidentes. Bafna *et al.*, 1997 [18] ont imaginé un algorithme pour ce problème avec un taux d'approximation de $2 - \frac{l}{k}$.

Les biologistes représentent souvent les similitudes entre deux séquences sous la forme de *matrices de points*. Une matrice de points est une matrice binaire ; un 1 à la position (i, j) indique une similitude entre la i-ième position de la première séquence et la j-ième position de la seconde. Les critères de similitude varient : ils peuvent être purement combinatoires (par exemple, un appariement de longueur fixée avec au plus k mésappariements qui commence en position i dans la première séquence et j dans la seconde) ou utiliser les coefficients de corrélation entre les paramètres physiques des acides aminés. Cependant, aucun critère n'est parfait dans sa capacité à distinguer les similitudes réelles (ayant un intérêt biologique) et les similitudes dues au hasard (bruits). Dans les applications biologiques, les bruits cachent les vraies similitudes et le problème est de savoir comment filtrer les bruits des matrices de points.

L'accès à quelques séquences comportant des similitudes d'origine biologique aide au filtrage des bruits. Lorsque k séquences sont données, on peut calculer C_k^2 matrices de points par paires. Si les k séquences ont une région de similitude commune, alors cette région doit être visible dans les C_k^2 matrices de points. Par ailleurs, un bruit a peu de chance d'apparaître uniformément parmi toutes les matrices de points. Le problème pratique est d'obtenir la réciproque : étant données les C_k^2 matrices de points, trouver les similitudes communes à

toutes ou presque toutes les séquences et séparer les bruits par filtrage. Vingron et Argos, 1991 [344] ont imaginé un algorithme permettant d'assembler une matrice de dimension k à partir de matrices de points de dimension 2.

7.2 Score d'un alignement multiple

Soit $\mathcal{A}$ un *alphabet* fini et soient $a_1, \ldots, a_k$ k séquences (chaînes) sur $\mathcal{A}$. Pour plus de commodité, on suppose que chacune de ces chaînes contient n caractères. Soit $\mathcal{A}' = \mathcal{A} \cup \{-\}$ un alphabet étendu, où le symbole « $-$ » désigne une espace. Un *alignement* de chaînes $a_1, \ldots, a_k$ est spécifié par une matrice A de taille $k \times m$, où m est un entier vérifiant $m \geq n$. Chaque élément de la matrice est un élément de $\mathcal{A}'$ et chaque ligne i contient les caractères de a_i dans l'ordre, séparés par $m - n$ espaces. On suppose également que chaque colonne de la matrice d'alignement multiple contient au moins un symbole de $\mathcal{A}$. Le score d'alignement multiple est défini comme la somme des scores des colonnes et l'alignement optimal est défini comme étant l'alignement qui minimise le score.

Le score d'une colonne peut être défini de nombreuses façons. La manière intuitive consiste à associer les plus grands scores aux colonnes ayant une grande variabilité de lettres. Par exemple, dans le problème de la *plus courte super-séquence multiple commune*, le score d'une colonne est défini comme le nombre de caractères distincts présents dans cette colonne. Dans le problème de la *plus longue sous-séquence multiple commune*, le score d'une colonne est -1 si tous les caractères de la colonne sont les mêmes et 0 sinon. Dans l'approche biologiquement plus pertinente de l'*entropie minimale*, le score des alignements multiples est défini comme la somme des entropies des colonnes. L'entropie d'une colonne i est définie par :

$$-\sum_{x \in \mathcal{A}'} p_x^i \ln(p_x^i),$$

où p_x^i est la fréquence de la lettre $x \in \mathcal{A}'$ dans la colonne i. Plus la colonne est variable, plus l'entropie est grande. Une colonne complètement conservée (comme dans le problème de la LCS multiple) doit avoir l'entropie minimale 0.

Le score d'entropie minimal correspond à la notion biologique de bon alignement, mais il est difficile à analyser efficacement en pratique. Les scores de la *distance de consensus* et de la *somme de paires* (SP) sont plus faciles à analyser.

- *Distance de consensus.* Le consensus d'un alignement est une chaîne formée des caractères les plus fréquents dans chaque colonne de l'alignement multiple. Le score de la *distance de consensus* est défini comme le nombre total de caractères de l'alignement qui diffèrent des caractères de consensus de leur colonne.

– *Somme de paires (score SP).* Pour un alignement multiple $A = (a_{ih})$, le score induit par l'alignement par paires A_{ij} pour les séquences a_i et a_j est

$$s(A_{ij}) = \sum_{h=1}^{m} d(a_{ih}, a_{jh}),$$

où d désigne la *distance* entre les éléments de $\mathcal{A}'$. Le *score de la somme de paires* (score SP) pour l'alignement A est donné par $\sum_{i,j} s(A_{ij})$. Dans cette définition, le score de l'alignement A est la somme des scores des *projections* de A sur toutes les paires de séquences a_i et a_j. La distance d possède les propriétés métriques, notamment $d(x,x) = 0$ et $d(x,z) \leq d(x,y) + d(y,z)$ pour tout triplet $(x,y,z) \in (\mathcal{A}')^3$.

7.3 Assemblage d'alignements par paires

Feng et Doolittle, 1987 [100] utilisent la paire de chaînes qui a la plus grande similitude et les « fusionnent » en une nouvelle chaîne suivant le principe « une brêche un jour, une brêche toujours ». Par suite, l'alignement multiple de k séquences se ramène à l'alignement multiple de $k-1$ séquences (l'une d'elles correspond aux chaînes fusionnées). On choisit des chaînes les plus proches lors des premières étapes de l'algorithme car des chaînes voisines fournissent l'information la plus fiable sur l'alignement.

Étant donnés un alignement A de séquences $a_1, \ldots, a_k$ et un alignement A' d'un sous-ensemble des séquences, on dit que A est *compatible* avec A' si A aligne les caractères des séquences qui sont alignés par A' de la même façon que A'. Feng et Doolittle, 1987 [100] ont observé la chose suivante : étant donné un arbre dans lequel chaque sommet est étiqueté avec une séquence distincte a_i et étant donnés des alignements par paires spécifiés pour chaque arête de l'arbre, il existe un alignement multiple des k séquences qui est compatible avec chacun des alignements par paires. En particulier, ce résultat est vrai pour une étoile à k sommets, c'est-à-dire un arbre possédant un sommet *central* et $k-1$ feuilles.

Lemme 7.1 *Pour toute étoile et tous les alignements par paires $A_1, \ldots, A_{k-1}$ spécifiés sur ses arêtes, il existe un alignement A pour les k séquences qui est compatible avec chacun des alignements $A_1, \ldots, A_{k-1}$.*

Étant donnée une étoile G, on définit un *alignement étoilé* A_G comme un alignement compatible avec les alignements optimaux par paires sur les arêtes de cette étoile. L'alignement A_G optimise $k-1$ parmi C_k^2 alignements par paires dans le score SP. La question de savoir à quel point l'alignement étoilé est bon est resté un problème ouvert jusqu'à ce que Gusfield, 1993 [144] prouve que, si l'étoile G est choisie convenablement, l'alignement étoilé est proche de l'alignement optimal avec un taux d'approximation de $2 - \frac{2}{k}$.

Soit $G(V, E)$ un graphe connexe (non orienté), soit $\gamma(i, j)$ une des plus courtes chaînes (fixée) entre les sommets $i \in V$ et $j \in V$ avec $i \neq j$, et soit $d(i, j)$ la longueur d'une telle plus courte chaîne. Pour une arête $e \in E$, on définit le coût de communication de e, noté $c(e)$, comme le nombre de plus courtes chaînes $\gamma(i, j)$ dans G qui utilisent l'arête e. Par exemple, le coût de communication de toute arête d'une étoile à k sommets vaut $k - 1$. On définit alors $c(G)$, le coût de communication du graphe G, de la façon suivante : $c(G) = \sum_{e \in E} c(e) = \sum_{i \neq j \in V} d(i, j)$. Le *graphe complet* à k sommets K_k a un coût de communication minimal de $c(G) = \frac{k(k-1)}{2}$ parmi tous les graphes connexes à k sommets. On appelle $b(G) = \frac{c(G)}{c(K_k)} = 2\frac{c(G)}{k(k-1)}$ le *coût de communication normalisé* de G. Pour une étoile à k sommets, on a : $b(G) = 2 - \frac{2}{k}$. Feng et Doolittle, 1987 [100] ont utilisé un arbre pour combiner les alignements par paires en un alignement multiple ; il s'est avéré que le coût de communication normalisé de cet arbre est lié au taux d'approximation de l'algorithme heuristique obtenu.

On définit $C(G) = (c_{ij})$ comme étant une matrice de taille $k \times k$, avec $c_{ij} = c(e)$ si $\{i, j\}$ est une arête e de G et $c_{ij} = 0$ sinon. Le *score SP pondéré* pour l'alignement A est :

$$\sum_{i,j} c_{ij} \times s(A_{ij}).$$

Pour simplifier les notations, $S(A) = (s(A_{ij}))$ désignant la matrice des scores des paires de séquences, on note $C(G) \odot S(A)$ le score SP pondéré. Soit E la matrice ayant des 0 sur sa diagonale principale et des 1 partout ailleurs ; le score SP (non pondéré) de l'alignement A est $E \odot S(A)$.

Les scores par paires d'un alignement héritent de la distance matricielle la propriété de l'inégalité triangulaire. Autrement dit, pour tout alignement A et pour tous i, j et k, on a : $s(A_{ij}) \leq s(A_{ik}) + s(A_{kj})$. On en déduit le lemme suivant :

Lemme 7.2 *Pour tout alignement A de k séquences et une étoile G, on a : $E \odot S(A) \leq C(G) \odot S(A)$.*

7.4 Algorithme d'approximation pour des alignements multiples

Soit $\mathcal{G}$ une collection d'étoiles dans un graphe à k sommets. On dit que cette collection est *équilibrée* s'il existe un réel $p > 1$ tel que l'on ait l'égalité $\sum_{G \in \mathcal{G}} C(G) = pE$. Par exemple, une collection de k étoiles avec k sommets centraux différents est une collection équilibrée avec $p = 2(k-1)$. Comme, pour $G \in \mathcal{G}$, les termes de $C(G)$ sont non nuls seulement en les arêtes de l'étoile G et comme l'alignement étoilé A_G induit des alignements optimaux sur les arêtes de G, on en déduit, pour tout alignement A :

$$C(G) \odot S(A_G) \leq C(G) \odot S(A).$$

Lemme 7.3 *Si $\mathcal{G}$ est un ensemble équilibré d'étoiles, il vérifie :*

$$\min_{G\epsilon\mathcal{G}} C(G) \odot S(A_G) \leq \frac{p}{|\mathcal{G}|} \min_{A} E \odot S(A).$$

Preuve On utilise un argument fondé sur le calcul de la moyenne :

$$\begin{aligned} \min_{G\in\mathcal{G}} C(G) \odot S(A_G) &\leq \tfrac{1}{|\mathcal{G}|} \textstyle\sum_{G\in\mathcal{G}} C(G) \odot S(A_G) \\ &\leq \tfrac{1}{|\mathcal{G}|} \times \textstyle\sum_{G\in\mathcal{G}} C(G) \odot S(A) = \tfrac{p}{|\mathcal{G}|} \times E \odot S(A). \end{aligned}$$

Ici, l'inégalité est vraie pour un alignement arbitraire A, donc, en particulier, pour un alignement optimal. ■

On définit l'algorithme **ALIGNER** de la façon suivante :

ALIGNER

1. Construire un ensemble $\mathcal{G}$ équilibré d'étoiles.

2. Pour chaque étoile G dans $\mathcal{G}$, assembler un alignement étoilé A_G.

3. Choisir une étoile G telle que $C(G) \odot S(A_G)$ soit minimal parmi toutes les étoiles dans $\mathcal{G}$.

4. Retourner A_G.

Théorème 7.1 *(Gusfield, 1993 [144]) Étant donnée une collection équilibrée d'étoiles $\mathcal{G}$,* **ALIGNER** *calcule un alignement avec une garantie de performance de $2 - 2/k$ en un temps $O(k \times n^2 \times |\mathcal{G}|)$.*

Preuve Il faut noter que l'on a : $\frac{p}{|\mathcal{G}|} = \frac{C(G) \odot E}{E \odot E} = 2 - \frac{2}{k}$. **ALIGNER** calcule l'alignement A_G qui est optimal pour une étoile $G \in \mathcal{G}$ et pour laquelle le plus petit score, $\min_{G\in\mathcal{G}} C(G) \odot S(A_G)$, est atteint. Les lemmes 7.2 et 7.3 impliquent l'inégalité suivante : $E \odot S(A_G) \leq C(G) \odot S(A_G) \leq \left(2 - \frac{2}{k}\right) \times \min_A E \odot S(A)$. ■

7.5 Assemblage de l-alignements

Une *l-étoile* $G = (V, E)$ à k sommets est définie par $r = \frac{k-1}{l-1}$ cliques de taille l ayant un seul sommet commun : le sommet *central* (figure 7.1). Une 3-étoile à $k = 2t+1$ sommets vérifie : $c(G) = (2t-1)2t+t$ et $b(G) = 2-\frac{3}{k}$. Pour une l-étoile à $k = (l-1)t+1$ sommets, on a : $c(G) = ((l-1)t+2-l)(l-1)t+t(\frac{l(l-1)}{2}-l+1)$

et $b(G) = 2 - \frac{l}{k}$. Le coût de communication d'une arête e dans une l-étoile G de centre c est :

$$c(e) = \begin{cases} k - l + 1, & \text{si } e \text{ est incidente à } c \\ 1, & \text{sinon.} \end{cases}$$

Il faut noter que, pour la matrice des coûts de communication d'une l-étoile G, on obtient :

$$C(G) \odot E = (k - l + 1) \times (k - 1) + \frac{k-1}{l-1} \times C_{l-1}^2 = C_k^2 \times \left(2 - \frac{l}{k}\right)$$

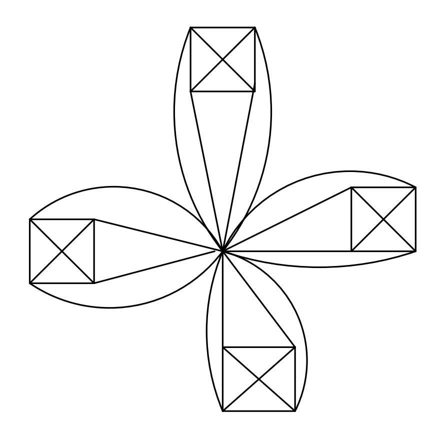


Figure 7.1 – Une 5-étoile avec quatre cliques ($l = 5$, $k = 17$).

Soient $A_1, \ldots, A_r$ des alignements pour les r cliques dans la l-étoile, où chaque A_i aligne l séquences. Une construction similaire à celle de Feng et Doolittle, 1987 [100] implique un lemme analogue au lemme 7.1 :

Lemme 7.4 *Pour toute l-étoile et pour tous les alignements $A_1, \ldots, A_r$ spécifiés pour ses cliques, il existe un alignement A pour les k séquences qui est compatible avec chacun des alignements $A_1, \ldots, A_r$.*

Il est possible de généraliser l'algorithme **Aligner** aux l-étoiles et de prouver des lemmes analogues aux lemmes 7.2 et 7.3. Par suite, le théorème 7.1 peut également être généralisé aux l-étoiles, menant ainsi à un algorithme dont la garantie de performance est égale au coût de communication normalisé des l-étoiles, qui vaut $2 - \frac{l}{k}$. La complexité temporelle de cet algorithme est en $O(k(2n)^l|\mathcal{G}|)$. Par conséquent, le problème consiste à trouver un petit ensemble de l-étoiles équilibré $\mathcal{G}$.

Nous avons ramené le problème de l'alignement multiple à celui qui consiste à trouver un alignement optimal pour chaque clique de chaque l-étoile dans un ensemble équilibré $\mathcal{G}$. À quel point est-il difficile de trouver un ensemble équilibré $\mathcal{G}$? Un candidat trivial est simplement constitué de toutes les l-étoiles ; par

symétrie, il est clairement équilibré. Pour $l = 2$, Gusfield, 1993 [144] a exploité le fait qu'il n'existe que k 2-étoiles pour construire un algorithme d'alignement multiple avec un taux d'approximation de $2 - \frac{2}{k}$. Il s'agit vraiment d'un cas particulier car, pour $l > 2$, le nombre de l-étoiles croît de façon exponentielle avec k ; cet algorithme étant fondé sur la formation de toutes les l-étoiles, il devient calculatoirement impossible à mettre en œuvre. Construire un *petit* ensemble équilibré de l-étoiles n'est pas trivial. Pevzner, 1992 [265] a résolu le cas $l = 3$ en cartographiant le problème de l'appariement maximal dans les graphes. Bafna *et al.*,1997 [18] ont ensuite proposé un algorithme de taux d'approximation $2 - \frac{l}{k}$ pour un l arbitraire.

7.6 Matrices de points et reconstruction d'images

Étant données k séquences, il est facile de produire C_k^2 matrices de points par paires. Il est beaucoup plus difficile de les assembler en une matrice de points k-dimensionnelle et de trouver des régions de similitude communes à toutes ou presque toutes les séquences. Pour aborder ce problème, Vihinen, 1988 [343] et Roytberg, 1992 [290] ont proposé de superposer les matrices de points par paires en choisissant une séquence de référence et en lui reliant toutes les autres. Ci-dessous, nous décrivons l'algorithme de Vingron et Argos, 1991 [344] pour l'assemblage d'une matrice de points k-dimensionnelle à partir de matrices de points bidimensionnelles.

On représente le problème de l'assemblage de similitudes par paires au moyen d'un treillis géométrique. On considère M points à coordonnées entières dans l'espace de dimension k :

$$(i_1^1, \ldots, i_k^1), \ldots, (i_1^M, \ldots, i_k^M),$$

dont on ne connaît pas les coordonnées. Supposons que l'on observe les projections de ces points sur chaque paire de coordonnées s et t, avec $1 \leq s < t \leq k$:

$$(i_s^1, i_t^1), \ldots, (i_s^M, i_t^M),$$

ainsi que certains autres points (des *bruits*). Supposons également que nous ne puissions pas distinguer les points représentant des projections « véritables » de ceux qui représentent des bruits. Le problème de la *reconstruction de l'image de dimension* k consiste à reconstruire M points k-dimensionnels à partir de C_k^2 projections données (avec bruits) sur les coordonnées s et t, pour $1 \leq s < t \leq k$.

Dans cette construction, chaque similitude (élément consensus) commune à k séquences biologiques correspond à un point à coordonnées entières $(i_1, \ldots i_k)$ dans l'espace de dimension k, où i_s est la coordonnée du consensus dans la s-ième séquence, $1 \leq s \leq k$. En pratique, il est difficile de trouver les points entiers $(i_1, \ldots, i_k)$ correspondant aux consensus. En revanche, il est facile de trouver (même avec beaucoup de bruits) les projections (i_s, i_t) de tous les consensus $(i_1, \ldots i_k)$ sur chaque paire de coordonnées s et t. Cette observation établit le

lien entre le problème de l'alignement multiple et celui de la reconstruction de l'image k-dimensionnelle.

À partir des points donnés dans les plans des côtés, on se propose de ne garder que ceux qui satisfont aux critères de cohérence suivants : le point (i, j) dans la projection s, t est dit *cohérent* si, pour toute autre dimension u, il existe un entier m tel que (i, m) appartient à la projection sur s et u, et (j, m) à la projection sur t et u (Gotoh, 1990 [135]). Évidemment, chaque point « véritable », c'est-à-dire chaque point provenant d'une projection d'un point en dimension k, est cohérent. En revanche, on souhaite que des points aléatoires représentant des bruits soient incohérents. Ce critère permet de séparer par filtrage la plupart des bruits (mais peut-être pas tous) et conduit à l'algorithme de Vingron et Argos, 1991 [344], qui multiplie et compare les matrices de points.

7.7 Alignement multiple *via* la multiplication de matrices de points

On modélise la collection de C_k^2 matrices de points sous la forme d'un *graphe k-parti* $G(V_1 \cup V_2 \cup \cdots \cup V_k, E)$, où V_i est l'ensemble des positions dans la i-ième séquence. On relie les sommets $i \in V_s$ et $j \in V_t$ par une arête e s'il existe un point à la position (i, j) de la matrice de points comparant les séquences s et t. Une arête $e \in E$ s'écrira $e = \{s, i|t, j\}$ si elle relie les sommets $i \in V_s$ et $j \in V_t$. Un *triangle* formé des trois arêtes $\{s, i|t, j\}, \{t, j|u, m\}$ et $\{s, i|u, m\}$ est noté $\{s, i|t, j|u, m\}$. On dit alors d'une arête $\{s, i|t, j\}$ qu'elle est *cohérente* si, pour tout (u, s, t) avec $u \notin \{s, t\}$ et $1 \leq u \leq k$, il existe un triangle $\{s, i|t, j|u, m\}$ pour un certain m. Un sous-ensemble $E' \subseteq E$ est dit *cohérent* si, pour toutes les arêtes $\{s, i|t, j\} \in E'$, il existe des triangles $\{s, i|t, j|u, m\}$, dont toutes les arêtes sont dans E', pour tout $u \in \{1, \ldots, k\}$, distinct de s et t. On dit que le graphe k-parti G est *cohérent* si l'ensemble de ses arêtes est cohérent. Il est clair que, si $G'(V, E')$ et $G''(V, E'')$ sont des graphes cohérents, alors leur *réunion* $G(V, E' \cup E'')$ est également cohérente. Par conséquent, on peut associer à chaque graphe k-parti un unique sous-graphe cohérent maximal. On s'intéresse au problème suivant :

Problème de la cohérence d'un graphe Trouver le sous-graphe cohérent maximal (au sens de l'inclusion) d'un graphe n-parti.

La matrice de points pour des séquences s et t est définie comme la matrice d'*adjacence* A_{st} :

$$(A_{st})_{ij} = \begin{cases} 1, & \text{si } \{s, i|t, j\} \in E \\ 0, & \text{sinon.} \end{cases}$$

Toute matrice de ce type correspond à un sous-ensemble de E ; on va appliquer aux matrices A_{st} les opérations $\cup$ (union), $\cap$ (intersection) et $\subset$ (inclusion). On reformule la définition précédente de la cohérence en termes de multiplication booléenne (que l'on note $\circ$) de matrices d'adjacence (Vingron et Argos,

1991 [344]). Un graphe k-parti est cohérent si et seulement si l'on a :

$$A_{st} \subseteq A_{su} \circ A_{ut} \text{ pour tout } (s,t,u), \text{ avec } 1 \leq s,t,u \leq k, s \neq t \neq u \text{ et } s \neq u. \quad (7.1)$$

La caractérisation (7.1) suggère d'utiliser la procédure simple suivante pour résoudre le problème de la cohérence : ne garder que les 1 de la matrice d'adjacence présents à la fois dans la matrice elle-même et dans tous ses produits $A_{su} \circ A_{ut}$. En faisant cela une fois pour toutes les matrices d'adjacence, on change également les matrices utilisées pour les produits. Ceci aboutit à l'algorithme de multiplication matricielle itérative qui part des matrices d'adjacence du graphe k-parti donné $A_{st}^{(0)} := A_{st}$ et qui définit la suite de matrices :

$$A_{st}^{(l+1)} := A_{st}^{(l)} \cap \left(\bigcap_{u \neq s,t} A_{su}^{(l)} \circ A_{ut}^{(l)} \right)$$

(les exposants permettent de distinguer les différentes itérations). Une fois que ceci a été fait pour tous les indices s et t, le processus se répète jusqu'à obtenir, pour une certaine itération, $A_{st}^{(l+1)} = A_{st}^{(l)}$ pour tous $1 \leq s \leq k, 1 \leq t \leq k$.

L'algorithme de multiplication de matrices de points (Vingron et Argos, 1991[344]) converge vers le sous-graphe cohérent maximal et chaque itération a une complexité temporelle en $O(L^3k^3)$, où L est la longueur des séquences. Comme le nombre d'itérations peut être vraiment très grand, Vingron et Pevzner, 1995 [345] ont imaginé un algorithme de complexité temporelle globale en $O(L^3k^3)$, ce qui équivaut à la complexité d'une seule itération de l'algorithme de multiplication matricielle. Dans les applications pratiques, les données de départ pour l'algorithme sont des matrices peu denses, ce qui rend l'algorithme encore plus rapide. Exprimée en fonction du nombre total M de points dans les C_k^2 matrices de points, la complexité temporelle est en $O(kLM)$.

7.8 Quelques autres problèmes et approches

7.8.1 Alignement multiple par arbres évolutifs

Il arrive souvent qu'en plus des séquences $a_1, \ldots, a_k$, les biologistes connaissent (ou supposent connue) l'histoire évolutive (représentée par un *arbre évolutif*) de ces séquences. Dans ce cas, on associe $a_1, \ldots, a_k$ aux feuilles de l'arbre et le problème consiste à reconstruire les séquences ancestrales (correspondant aux sommets internes de l'arbre) qui minimisent le nombre total de mutations sur les arêtes de l'arbre. Le score d'une arête de ce dernier est égal à la distance d'édition entre les séquences associées à ses extrémités ; le score de l'arbre évolutif est la somme des scores de toutes les arêtes de l'arbre. Pour un arbre évolutif donné, l'alignement multiple optimal est l'ensemble des séquences associées aux sommets internes de l'arbre qui produit le score minimal (Sankoff, 1975 [298]). Wang *et al.*, 1996 [352] et Wang et Gusfield, 1996 [350] ont

développé des algorithmes d'approximation avec garantie de performance pour l'alignement d'arbres évolutifs.

7.8.2 Coupure des coins dans les graphes d'édition

Carrillo et Lipman, 1988 [58] et Lipman *et al.*, 1989 [224] ont suggéré une méthode de séparation et d'évaluation progressive pour l'alignement multiple. En effet, si l'un des alignements par paires imposé par un alignement multiple est mauvais, alors l'alignement multiple global n'aura pas un bon score. Cette observation implique qu'un « bon » alignement multiple impose de « bons » alignements par paires, limitant ainsi une recherche au voisinage d'une diagonale principale dans une matrice d'alignement k-dimensionnelle.

Chapitre 8

Trouver des signaux dans l'ADN

8.1 Introduction

Le premier signal dans l'ADN a peut-être été trouvé en 1970 par Hamilton Smith, après la découverte de l'enzyme de restriction *Hin*dII. Le site palindromique de l'enzyme de restriction est un signal qui initie la coupure de l'ADN. Trouver la séquence de ce site n'était pas simple en 1970. En fait, Hamilton Smith a publié deux articles consécutifs sur *Hin*dII : un sur la purification de l'enzyme et l'autre sur la découverte du signal de reconnaissance de l'enzyme (Kelly et Smith, 1970 [196]).

En revenant au début des années 70, on réalise que Hamilton Smith a eu de la chance : les sites de restriction sont les plus simples à détecter dans l'ADN. Trente ans plus tard, ils demeurent peut-être les seuls signaux que l'on puisse trouver dans l'ADN de façon certaine. La plupart des autres signaux (promoteurs, sites de raccordement, etc.) sont tellement compliqués que l'on ne dispose pas encore de bon modèle ni d'algorithme fiable permettant de les reconnaître.

La compréhension de la régulation génique est un défi majeur en bio-informatique. Par exemple, la régulation de l'expression génique peut impliquer que le fait qu'une protéine se lie à une région d'ADN affecte la transcription d'un gène adjacent. Comme les mécanismes d'interaction protéine-ADN sont encore insuffisamment compris pour permettre la prédiction *in silico* des sites d'agglomération, l'approche expérimentale courante consiste à localiser la position approximative de ces sites. Ces expériences mènent habituellement à l'identification d'un fragment d'ADN de longueur n qui contient un site agglomérant (un *mot magique* inconnu) de longueur $l \ll n$. Une telle expérience est insuffisante pour trouver le site agglomérant, mais un échantillon du fragment d'ADN trouvé expérimentalement donne un espoir de trouver le mot magique.

Dans sa forme la plus simple, le problème de la découverte du signal (et le problème du site de l'enzyme de restriction en particulier) peut être formulé de la façon suivante. Supposons qu'on nous donne un échantillon de K séquences et supposons qu'un mot magique (inconnu) apparaisse à différentes positions (inconnues) de ces séquences. Peut-on trouver le mot magique ?

Une approche de bon sens au problème du mot magique consiste à tester tous les mots de longueur l et de trouver ceux qui apparaissent dans toutes (ou presque toutes) les séquences de l'échantillon (Staden, 1989 [326], Wolfertstetter *et al.*, 1996 [370] et Tompa, 1999 [338]). Si le mot magique est le seul mot qui apparaît aussi fréquemment dans l'échantillon, alors le problème est (probablement) résolu. Si ce n'est pas le cas, il faut augmenter l et répéter la procédure.

L'approche décrite fonctionne habituellement bien pour des mots continus courts comme $GAATTC$, le site de restriction de EcoRI. Cependant, si la longueur des séquences dans l'échantillon avoisine 4^6, des mots aléatoires peuvent commencer à faire concurrence au mot magique, car certains d'entre eux peuvent apparaître dans de nombreuses séquences, simplement par hasard. La situation devient encore plus difficile si les fréquences des nucléotides dans l'échantillon ont une distribution biaisée.

Le problème se complique encore lorsque le mot magique présente des trous, comme dans $CCAN_9TGG$, le site de l'enzyme de restriction Xcm I (N désigne n'importe quel nucléotide et N_9 indique un trou de longueur 9 dans le site). Évidemment, on peut essayer d'énumérer tous les modèles avec trous, mais la complexité temporelle du problème croît très rapidement, en particulier si l'on autorise des modèles avec beaucoup de trous. Même trouver le mot magique relativement simple « lettre-trou-lettre-trou-lettre » n'est plus si facile que ça ; il a au moins justifié un autre article sur la découverte de modèles par Hamilton Smith et ses collègues vingt ans après la découverte du modèle du premier site de restriction (Smith *et al.*, 1990 [318]). Une autre tâche de reconnaissance intimidante est de trouver des signaux comme $Pu^mCN_{40-2000}Pu^mC$, le site de reconnaissance de l'endonucléase McrBC (Pu désigne A ou G).

Alors que les problèmes évoqués ne sont pas simples, les vrais problèmes biologiques dans la découverte de signaux sont encore beaucoup plus compliqués. Les signaux biologiques peuvent être longs, troués et flous. Par exemple, $TTGACAN_{17}TATAAT$ est le mot magique pour les promoteurs de *E.coli*. L'énumération et la vérification de tous les modèles de ce type sont difficilement possibles en raison de la complexité calculatoire. Cependant, même si l'on énumérait tous les modèles de ce type, cela ne nous aiderait que très peu car le modèle ci-dessus représente un promoteur *idéal* qui n'apparaît jamais dans les séquences des promoteurs connus. Il s'agit plutôt d'un consensus de tous les promoteurs connus : ni les bases de consensus, ni l'espacement entre les deux parties du signal ne sont conservés. En d'autres termes, la description du mot magique dans ce cas donne quelque chose comme « douze positions non dégénérées avec un trou et un maximum de quatre mésappariements ». Il n'existe pas encore d'algorithme fiable pour trouver ce type de signal. Le défaut des

algorithmes existants est que, pour des signaux subtils, ils convergent souvent vers des minima locaux qui ne représentent pas la solution cherchée.

8.2 Edgar Allan Poe et la linguistique de l'ADN

Lorsque, dans la nouvelle d'Edgar Allan Poe intitulée « Le scarabée d'or », William Legrand trouva un parchemin écrit par le pirate Capitaine Kidd,

5 3 ‡‡†3 0 5)) 6 *; 4 8 2 6) 4 ‡.) 4 ‡) : 8 0 6 *; 4 8 †8 ¶6 0)) 8
5; 1 ‡(; : ‡* 8 †8 3 (8 8) 5 * †; 4 6 (8 8 * 9 6 *?; 8) * ‡(; 4 8 5
); 5 * †2 : * ‡(; 4 9 5 6 * 2 (5 * - - 4) 8 ¶8 *; 4 0 6 9 2 8 5);) 6 †8
) 4 ‡‡; 1 (‡9; 4 8 0 8 1; 8 : 8 ‡1; 4 8 †8 5; 4) 4 8 5 †5 2 8 8 0 6
* 8 1 (‡9; 4 8; (8 8; 4 (‡? 3 4; 4 8) 4 ‡; 1 6 1; : 1 8 8; ‡?;

son ami lui dit : « Si tous les trésors de Golconde devaient être pour moi le prix de la solution de cette énigme, je serais parfaitement sûr de ne pas les gagner ». Mr. Legrand lui répondit la chose suivante : « Il est vraiment douteux que l'ingéniosité humaine puisse créer une énigme de ce genre, dont l'ingéniosité humaine ne vienne à bout par une application suffisante ». Il nota qu'une combinaison de trois symboles — ; 4 8 — apparaissait très fréquemment dans le texte. Il savait également que les pirates du Capitaine Kidd parlaient l'anglais et que le mot anglais le plus fréquent est le mot « the ». En supposant que ; 4 8 codait « the », Mr. Legrand déchiffra la note du parchemin et trouva le trésor des pirates (ainsi que quelques squelettes). Avec cet indice, Mr. Legrand disposait d'un texte légèrement plus facile à déchiffrer :

5 3 ‡‡†3 0 5)) 6 * T H E 2 6) H ‡.) H ‡) : E 0 6 * T H E †E ¶6 0)) E
5 T 1 ‡(T : ‡* E †E 3 (E E) 5 * †T 4 6 (E E * 9 6 * ? T E) * ‡(T H E 5
) T 5 * †2 :*‡(T H 9 5 6 * 2 (5 * - - H) E ¶E * T H 0 6 9 2 E 5) T) 6 †E
) H ‡‡T 1(‡9 T H E 0 E 1 T E :E ‡1 T H E †E 5 T H)H E 5†5 2 E E 0 6
*E 1(‡9 T H E T (E E T H (‡ ? 3 H T H E) H‡T 1 6 1 T :1 E E T ‡ ? T

Vous pouvez essayer de deviner ce que code le symbole « (» et compléter le déchiffrage.

Les textes d'ADN ne sont pas faciles à déchiffrer et on se demande si la nature a pu créer une énigme de ce genre dont l'ingéniosité humaine ne vienne à bout. Cependant, la linguistique de l'ADN a emprunté la méthode scientifique de Mr. Legrand et il existe une approche classique dans la linguistique de l'ADN, fondée sur l'hypothèse selon laquelle des mots fréquents ou rares peuvent correspondre à des signaux dans l'ADN. Si un mot apparaît considérablement plus (ou moins) fréquemment que ce à quoi on pouvait s'attendre, alors il devient un « signal » potentiel et l'on se pose la question de la signification « biologique » de ce mot (Brendel *et al.*, 1986 [47] et Burge *et al.*, 1992 [53]). Par exemple, Gelfand et Koonin, 1997 [124] ont montré que les 6-palindromes les moins présents dans l'archéon *M.jannaschii* ont des chances d'être le site de reconnaissance d'un système de restriction-modification.

La linguistique de l'ADN est au cœur de l'approche *par modélisation* de la découverte de signaux ; elle se fonde sur l'énumération de tous les modèles possibles, parmi lesquels on choisit le plus fréquent ou le plus convenable (Brazma *et al.*, 1998 [46]). Les mesures d'ajustement vont des estimations de la significativité statistique des signaux découverts, à l'information contenue dans les fragments correspondant approximativement au signal. L'approche par modélisation inclut les étapes suivantes :

- Définir la mesure d'ajustement (la fréquence, par exemple).
- Calculer l'ajustement de chaque mot relativement à un échantillon de fragments d'ADN.
- Considérer les mots les mieux ajustés comme des signaux potentiels.

L'efficacité est l'un des problèmes de l'approche par modélisation, car l'espace de recherche pour des modèles de longueur l est $|\mathcal{A}|^l$, où $\mathcal{A}$ désigne l'alphabet. Pour élaguer la recherche, on peut utiliser l'idée qui sous-tend l'algorithme de Karp-Miller-Rosenberg (Karp *et al.*, 1972 [188]) : si une chaîne apparaît dans k séquences, alors toutes ses sous-chaînes apparaissent dans au moins k séquences. Toute chaîne fréquente peut donc être assemblée à partir de sous-chaînes fréquentes. Pour faire cela simplement, on peut créer une liste de tous les $2l$-uplets fréquents à partir de la liste de tous les l-uplets fréquents, en concaténant chaque paire de ses éléments et en vérifiant ensuite la fréquence de ces concaténations. Une autre approche consiste à utiliser les *arbres de suffixes* (Gusfield, 1997 [145]).

Pour trouver des mots fréquents et des mots rares dans un texte, il faut calculer la valeur moyenne (l'espérance) $E(W)$ et la variance $\sigma^2(W)$ du nombre d'occurrences (la fréquence) de chaque mot W. Ensuite, on identifie les mots fréquents et les mots rares comme étant ceux qui dévient de façon significative des fréquences moyennes. Dans de nombreux articles sur la linguistique de l'ADN, la variance $\sigma^2(W)$ du nombre d'occurrences d'un mot dans un texte a été supposée égale à $E(W)$ par erreur.

Trouver la probabilité de k occurrences d'un mot dans un texte nécessite le recours aux fonctions génératrices et à l'analyse complexe (Guibas et Odlyzko, 1981 [141]). La probabilité de l'occurrence d'un mot dans un texte ne dépend pas seulement de la longueur de ce mot, mais également de la structure de ses chevauchements, définie par le *polynôme d'autocorrélation* (Guibas et Odlyzko, 1981 [141]). Par exemple, la distribution du nombre d'occurrences de AAA (de polynôme d'autocorrélation $1 + x + x^2$) diffère de façon significative de la distribution du nombre d'occurrences de ATA (de polynôme d'autocorrélation $1+x^2$), même dans un texte aléatoire de Bernoulli avec des lettres équiprobables (*paradoxe des mots qui se chevauchent*). Ci-dessous, nous discutons du meilleur pari pour les naïfs, afin d'expliquer ce paradoxe.

8.3 Meilleur pari pour les naïfs

Le paradoxe des mots qui se chevauchent est la base du *meilleur pari pour les naïfs*, étudié par John Conway. Ce jeu commence avec deux joueurs I et II qui choisissent des mots de longueur l dans un alphabet 0-1. Le joueur I choisit une suite A de l piles ou faces et le joueur II, qui connaît A, choisit une autre suite B de longueur l. Les joueurs jouent ensuite à pile ou face, jusqu'à ce que A ou B apparaisse sous la forme d'un bloc de l résultats consécutifs. Le jeu se termine avec la probabilité 1.

Une nuit, à Elkhorn, le Courtaud demanda à Belliou-la-Fumée : « C'est drôle que vous ne jouiez pas d'argent, n'est-ce pas dans votre nature ? » (dans « Belliou-la-Fumée », de Jack London). Belliou lui répondit : « C'est vrai. Mais j'ai les statistiques en tête. J'aime que les chances soient égales lorsque je joue mon argent ».

À première vue, il semblerait que A et B aient la même chance de sortir. Même si quelqu'un réalise que certains mots sont plus probables que d'autres dans ce jeu, on a l'impression que A devrait gagner après avoir choisi le mot « le plus fort ». L'aspect mystérieux de ce jeu tient au fait que, pour l supérieur ou égal à 3, peu importe ce que vaut A, le joueur II peut toujours choisir un mot B qui bat A. Le plus surprenant est que le meilleur pari pour les naïfs est un jeu non transitif : A bat B et B bat C n'impliquent pas que A bat C (pensez au jeu pierre-feuille-ciseaux !).

Supposons que le joueur I choisisse 00 et que le joueur II prenne 10. Après deux lancers, soit I gagne (00), soit II gagne (10), soit le jeu continue (01 ou 11). Cependant, le joueur I n'a plus aucun intérêt à continuer, car l'autre va gagner, quoi qu'il arrive ! Par conséquent, la cote de B contre A dans ce jeu est de 3 contre 1.

L'analyse du meilleur pari pour les naïfs repose sur la notion de *polynôme de corrélation* (Guibas et Odlyzko, 1981 [141]). Étant donnés deux mots de l lettres A et B, la *corrélation* de A et B, notée $AB = (c_0, \ldots, c_{l-1})$, est un mot booléen de l lettres (figure 8.1). Par définition, le i-ième bit de AB vaut 1 si le $(n-i)$-préfixe (les $n-i$ premières lettres) de B coïncide avec le $(n-i)$-suffixe (les $n-i$ dernières lettres) de A. Sinon, le i-ième bit de AB vaut 0. Par définition, le *polynôme de corrélation* de A et B est $K_{AB}(t) = c_0 + c_1 \times t^1 + \ldots + c_{l-1} \times t^{l-1}$. On note encore $K_{AB} = K_{AB}(\frac{1}{2})$.

John Conway a suggéré l'élégante formule qui suit pour calculer la probabilité que B gagne contre A :

$$\frac{K_{AA} - K_{AB}}{K_{BB} - K_{BA}}$$

Conway n'a jamais publié la preuve de cette formule. Martin Gardner, 1974 [118] a écrit à propos de cette formule : « Je n'ai aucune idée de la raison pour laquelle elle fonctionne. Elle donne juste la réponse comme par magie, comme tant d'autres algorithmes de Conway. »

		AB
A= XYYXYY		
B= YYXYYX	*décalage* = 0	0
YYXYYX	*décalage* = 1	1
YYXYYX	*décalage* = 2	0
YYXYYX	*décalage* = 3	0
YYXYYX	*décalage* = 4	1
YYXYYX	*décalage* = 5	1

$$K_{AB} = t^1 + t^4 + t^5$$

$$H_{AB} = \{X, XYYX, XYYXY\}$$

$$K_{AB}(1/2) = 1/2 + 1/16 + 1/32 = P(H_{AB})$$

Figure 8.1 – Le polynôme de corrélation des mots A et B ($AB = (010011)$) et l'ensemble $\mathcal{H}_{AB}$.

Des preuves de cette formule ont été données indépendemment par Li, 1980 [222] à l'aide de martingales et par Guibas et Odlyzko, 1981 [141] avec des fonctions génératrices. Dans la suite, on donne une courte preuve de l'équation de Conway (Pevzner, 1993 [268]).

8.4 Équation de Conway

Soit $AB = (c_0, \ldots, c_{l-1})$ une corrélation de A et B et soient $c_{m_1}, \ldots, c_{m_k}$ les bits de AB égaux à 1. On note $\mathcal{H}_{AB}$ l'ensemble des k préfixes de $A = a_1 \ldots a_l$ de longueurs $m_1, \ldots, m_k$ (figure 8.1) :

$$(a_1 \ldots a_{m_1}),\ (a_1 \ldots a_{m_1} \ldots a_{m_2}),\ \ldots,\ (a_1 \ldots a_{m_1} \ldots a_{m_2} \ldots \ldots a_{m_k}).$$

Étant donnés deux mots X et Y, on note $X * Y$ la *concaténation* de X et Y. Étant donnés deux ensembles de mots $\mathcal{X} = \{X\}$ et $\mathcal{Y} = \{Y\}$, on note $\mathcal{X} * \mathcal{Y}$

l'ensemble de toutes les concaténations des mots de $\mathcal{X}$ et de $\mathcal{Y}$. L'ensemble $\mathcal{X} * \mathcal{Y}$ contient $|\mathcal{X}| \times |\mathcal{Y}|$ mots (avec d'éventuelles répétitions).

On note $P(X)$ la probabilité qu'un mot booléen X à l lettres représente le résultat de l lancers d'une pièce. On vérifie l'égalité $P(X) = \frac{1}{2^l}$. Pour un ensemble de mots $\mathcal{X} = \{X\}$, on note :

$$P(\mathcal{X}) = \sum_{X \epsilon \mathcal{X}} P(X).$$

Nous allons utiliser le résultat suivant :

Lemme 8.1 $K_{AB}(\frac{1}{2}) = P(\mathcal{H}_{AB})$.

Un mot W est une A-*victoire* s'il contient A à la fin et qu'il ne contient pas B. Un mot W est une A-*prévictoire* si $W * A$ est une A-victoire. On définit $\mathcal{S}_A$ comme étant l'ensemble de toutes les A-prévictoires. Les B-*victoires*, les B-*prévictoires* et l'ensemble $\mathcal{S}_B$ formé de toutes les B-prévictoires sont définis de façon semblable.

L'idée centrale de la preuve du lemme consiste à considérer tous les mots *non victorieux* :

$$\mathcal{T} = \{T : \ T \text{ n'est ni } A\text{-victorieux, ni } B\text{-victorieux}\}.$$

Tout mot $T * A$, pour $T \in \mathcal{T}$, correspond soit à une A-victoire, soit à une B-victoire. Si $T * A$ correspond à une A-victoire, alors T peut s'écrire sous la forme : A-prévictoire * H_{AA}, où $H_{AA} \in \mathcal{H}_{AA}$ (figure 8.2a). Si $T * A$ est une B-victoire, alors T peut être représenté sous la forme : B-prévictoire * H_{BA}, où $H_{BA} \in \mathcal{H}_{BA}$ (figure 8.2b). Ceci implique la représentation suivante pour les non-victoires :

Lemme 8.2 $\mathcal{T} = \mathcal{T}_1 = (\mathcal{S}_B * \mathcal{H}_{BA}) \cup (\mathcal{S}_A * \mathcal{H}_{AA})$.

De la même façon, tout mot $T * B$, pour $T \in \mathcal{T}$, correspond soit à une A-victoire, soit à une B-victoire. Si $T * B$ correspond à une A-victoire, alors T peut s'écrire sous la forme : A-prévictoire * H_{AB}, où $H_{AB} \in \mathcal{H}_{AB}$ (figure 8.2c). Si $T * B$ correspond à une B-victoire, alors T peut être représenté sous la forme : B-prévictoire * H_{BB}, où $H_{BB} \in \mathcal{H}_{BB}$ (figure 8.2d). Ceci implique une autre représentation des non-victoires :

Lemme 8.3 $\mathcal{T} = \mathcal{T}_2 = (\mathcal{S}_A * \mathcal{H}_{AB}) \cup (\mathcal{S}_B * \mathcal{H}_{BB})$.

Théorème 8.1 *La probabilité que B gagne contre A sont de* $\frac{K_{AA} - K_{AB}}{K_{BB} - K_{BA}}$.

Preuve Les lemmes 8.1 et 8.3 impliquent que la probabilité globale des mots dans l'ensemble $\mathcal{T}_2$ est :

$$\begin{aligned} P(\mathcal{T}_2) &= P(\mathcal{S}_A * \mathcal{H}_{AB}) + P(\mathcal{S}_B * \mathcal{H}_{BB}) \\ &= P(\mathcal{S}_A) \times P(\mathcal{H}_{AB}) + P(\mathcal{S}_B) \times P(\mathcal{H}_{BB}) \\ &= P(\mathcal{S}_A) \times K_{AB} + P(\mathcal{S}_B) \times K_{BB}. \end{aligned}$$

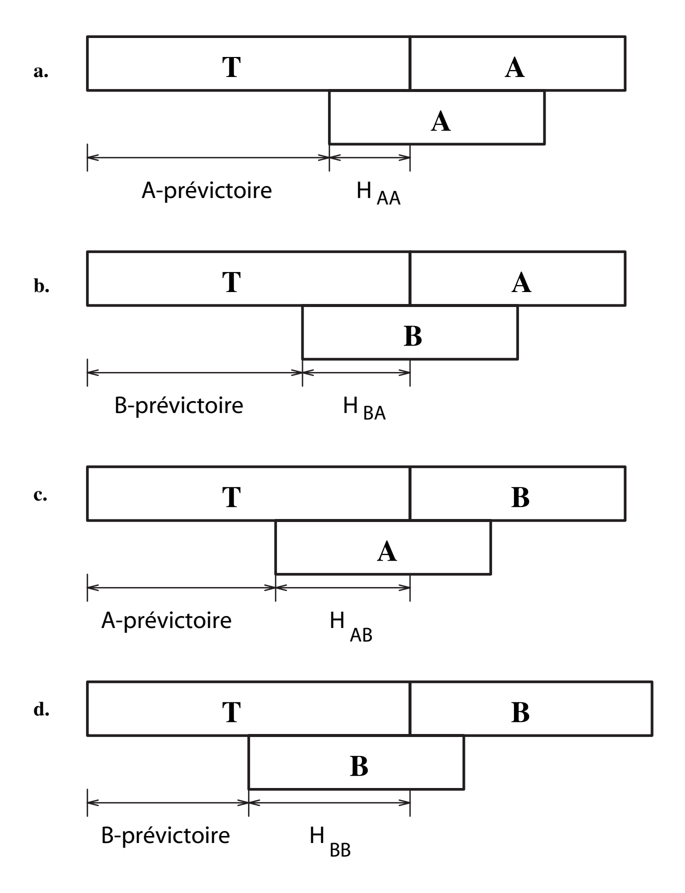


Figure 8.2 – Différentes représentations possibles pour les mots des ensembles $\mathcal{T} * \{A\}$ et $\mathcal{T} * \{B\}$.

De la même façon, les lemmes 8.1 et 8.2 impliquent :

$$P(\mathcal{T}_1) = P(\mathcal{S}_B) \times K_{BA} + P(\mathcal{S}_A) \times K_{AA}.$$

D'après les lemmes 8.2 et 8.3, $\mathcal{T}_1$ et $\mathcal{T}_2$ représentent le même ensemble $\mathcal{T}$; par conséquent, on a $P(\mathcal{T}_1) = P(\mathcal{T}_2)$ et donc :

$$P(\mathcal{S}_A) \times K_{AB} + P(\mathcal{S}_B) \times K_{BB} = P(\mathcal{S}_B) \times K_{BA} + P(\mathcal{S}_A) \times K_{AA}.$$

Ceci implique l'égalité :

$$\frac{P(\mathcal{S}_B)}{P(\mathcal{S}_A)} = \frac{K_{AA} - K_{AB}}{K_{BB} - K_{BA}}. \qquad \blacksquare$$

8.5 Mots fréquents dans l'ADN

Les formules pour la variance $Var(W)$ du nombre d'occurrences d'un mot W dans un texte de Bernoulli ont été données par Gentleman et Mullin, 1989 [126] et Pevzner *et al.*, 1989 [269]. Pour un texte de Bernoulli de longueur n dans un alphabet de l lettres avec des lettres équiprobables, on a :

$$Var(W) = \frac{n}{l^k} \times \left(2 \times K_{WW}(\frac{1}{l}) - 1 - \frac{2k-1}{l^k}\right),$$

où k est la longueur du mot W et $K_{WW}(t)$ le *polynôme d'autocorrélation de W* (c'est-à-dire le polynôme de corrélation des mots W et W).

Pour trouver cette formule, on considère (pour simplifier) un texte circulaire de longueur n dans un alphabet de l lettres, où la probabilité de chaque lettre à chaque position vaut $\frac{1}{l}$. Pour un mot W de k lettres, on définit une variable aléatoire x_i : elle vaut 1 si W commence à la i-ième position du texte et 0 sinon. On note $p = \frac{1}{l^k}$ l'espérance de x_i. Le nombre d'occurrences de W dans le texte est donné par la variable aléatoire :

$$X = \sum_{i=1}^{n} x_i,$$

d'espérance :

$$E(X) = \sum_{i=1}^{n} E(x_i) = np$$

et de variance :

$$Var(X) = E(X^2) - E(X)^2 = \sum_{\{1 \le i,j \le n\}} E(x_i x_j) - E(x_i)E(x_j).$$

Soit $d(i,j)$ la (plus courte) distance entre les positions i et j dans un texte circulaire. On obtient :

$$\begin{aligned} Var(X) &= \sum_{\{(i,j):\ d(i,j)\geq k\}} E(x_ix_j) - E(x_i)E(x_j) \\ &+ \sum_{\{(i,j):\ d(i,j)=0\}} E(x_ix_j) - E(x_i)E(x_j) \\ &+ \sum_{\{(i,j):\ 0<d(i,j)<k\}} E(x_ix_j) - E(x_i)E(x_j). \end{aligned}$$

Comme les variables aléatoires x_i et x_j sont indépendantes pour $d(i,j) \geq k$, le premier terme dans la formule ci-dessus est nul ; le deuxième est simplement égal à $n(p-p^2)$ et le dernier peut être réécrit de la façon suivante :

$$\sum_{\{(i,j):\ 0<d(i,j)<k\}} E(x_ix_j) - E(x_i)E(x_j) = \sum_{i=1}^{n}\sum_{t=1}^{k-1}\sum_{\{j:\ d(i,j)=t\}} E(x_ix_j) - E(x_i)E(x_j)$$

Pour un t fixé, $E(x_ix_{i+t})$ vaut $p\frac{1}{l^t}$ si le t-ième coefficient de la corrélation WW vaut 1 ; sinon, il est nul. On peut donc écrire $E(x_ix_{i+t}) = c_t p \frac{1}{l^t}$ et, comme il existe exactement deux positions j pour chaque i avec $d(i,j) = t$, on a :

$$\begin{aligned} \sum_{t=1}^{k-1}\sum_{\{j:\ d(i,j)=t\}} E(x_ix_j) &= 2p\sum_{t=1}^{k-1} c_t \frac{1}{l^t} \\ &= 2p(K_{WW}(\frac{1}{l}) - 1). \end{aligned}$$

Par conséquent, on obtient :

$$\begin{aligned} \sum_{\{(i,j):\ 0<d(i,j)<k\}} E(x_ix_j) - E(x_i)E(x_j) &= \sum_{i=1}^{n}\left(2p(K_{WW}(\frac{1}{l}) - 1) - 2(k-1)p^2\right) \\ &= np\left(2K_{WW}(\frac{1}{l}) - 2 - 2(k-1)p\right) \end{aligned}$$

et :

$$Var(X) = np\left(2K_{WW}(\frac{1}{l}) - 1 - (2k-1)p\right).$$

Ce résultat démontre que la variance de la fréquence des occurrences varie de façon significative entre les mots, même pour des textes de Bernoulli. En particulier, pour un alphabet de quatre lettres avec des probabilités égales pour les lettres A, T, G et C, on a :

$$\frac{Var(AA)}{Var(AT)} = \frac{21}{13} \quad \text{et} \quad \frac{Var(AAA)}{Var(ATG)} = \frac{99}{59}.$$

Par conséquent, ignorer le paradoxe des mots qui se chevauchent amène environ 200% d'erreurs dans les estimations de la significativité statistique, lors de l'analyse des mots fréquents. Pour des alphabets de deux lettres Pur/Pyr ou S/W, le fait d'ignorer ce paradoxe amène environ 500% d'erreurs dans les estimations de la significativité statistique ($\frac{Var(SS)}{Var(SW)} = \frac{5}{1}$) !

Les formules ci-dessus nous permettent de calculer la variance pour des textes de Bernoulli. Fousler et Karlin, 1987 [112], Stuckle *et al.*, 1990 [330] et Kleffe et Borodovsky, 1992 [199] ont présenté des formules approchées permettant de calculer la variance pour des textes engendrés par des chaînes de Markov. Prum *et al.*, 1995 [281] ont obtenu la distribution normale limite pour le nombre d'occurrences d'un mot dans le modèle markovien. Enfin, Regnier et Szpankowski, 1998 [282] ont étudié les occurrences approximatives de mots et en ont déduit les formules exactes et asymptotiques pour l'espérance, la variance et la probabilité des occurrences approchées.

8.6 Analyse des mots consensus

Pour un mot W et un échantillon $\mathcal{S}$, on note $W(\mathcal{S})$ le nombre de séquences de $\mathcal{S}$ qui contiennent W. Si un mot magique apparaît de façon exacte dans l'échantillon, alors un simple comptage de $W(\mathcal{S})$ pour chaque mot W de l lettres détecte le mot magique comme étant le plus fréquent. Le problème devient plus compliqué lorsque l'on autorise jusqu'à k erreurs (autrement dit, des mésappariements) dans le mot magique. Dans ce cas, Waterman *et al.*, 1984 [358] et Galas *et al.*, 1985 [115] ont suggéré d'analyser les *mots consensus* ; ceci consiste essentiellement en un comptage de mots approximatif. Pour chaque mot W, on définit un voisinage constitué de tous les mots situés à une distance inférieure ou égale à k de W, puis on compte les occurrences des mots de ce voisinage dans l'échantillon. On introduit également l'idée d'*occurrences pondérées* et on associe un poids plus grand aux voisins ayant le moins d'erreurs. Grâce à l'analyse des mots consensus, Galas *et al.*, 1985 [115] ont pu détecter les séquences consensus $TTGACA$ et $TATAAT$ dans le signal promoteur de *E. coli.*

Soit $D_H(s,t)$ la distance de Hamming entre deux chaînes s et t de même longueur. Mirkin et Roberts, 1993 [238] ont montré que le comptage de mots approximatif équivaut, en un certain sens, au problème suivant :

Problème de la chaîne consensus Étant donné un échantillon $\mathcal{S} = \{s_1, \ldots, s_n\}$ de séquences et un entier l, trouver une chaîne *médiane* s de longueur l et une sous-chaîne t_i de longueur l de chaque s_i qui minimisent $\sum_{i=1}^{n} d_H(s, t_i)$.

Li *et al.*, 1999 [221] ont montré que le problème de la chaîne consensus est NP-complet et ont donné un système d'approximation en un temps polynomial (PTAS) pour ce problème. L'algorithme repose sur la notion de chaîne *majoritaire*. Étant donnée une collection $t_1, \ldots, t_n$ de n chaînes de longueur l, la

chaîne majoritaire pour $t_1, \ldots, t_n$ est la chaîne s dont la i-ième lettre est la plus fréquente parmi les n i-ièmes lettres de $t_1, \ldots, t_n$. Li *et al.*, 1999 [221] ont inventé un PTAS pour le problème de la chaîne consensus qui est fondé sur le choix d'une chaîne majoritaire pour chacune des r sous-chaînes de longueur l de $\{s_1, \ldots, s_n\}$, $t_{i_1}, \ldots, t_{i_r}$.

Il est souvent pratique de concaténer des séquences multiples à partir d'un échantillon $\mathcal{S}$ en une seule séquence composée. On transforme alors le problème qui consiste à trouver la chaîne consensus en celui qui revient à découvrir la chaîne la plus fréquente dans le texte. Une approche assez naïve consiste à trouver le nombre d'occurrences $W(T)$ de chaque chaîne W de l lettres dans le texte T. Apostolico et Preparata, 1996 [10] ont imaginé un algorithme efficace pour le problème suivant :

Problème statistique des chaînes Étant donnés un texte T et un entier l, trouver $W(T)$ pour chaque chaîne W de l lettres.

Le problème statistique des chaînes devient difficile si l'on considère les occurrences de chaînes approximatives. Soit $W_k(T)$ le nombre d'occurrences approximatives de W dans T ayant jusqu'à k mésappariements. On ne connaît pas d'algorithme efficace qui permette de résoudre le problème suivant :

Problème de la chaîne fréquente Étant donnés un texte T et des entiers l et k, trouver une chaîne W de l lettres qui maximise $W_k(T)$ parmi tous les mots de l lettres.

L'analyse des mots consensus est un exemple d'approche par *séquençage* pour la découverte de signaux. Une approche par séquençage n'énumère pas tous les modèles : elle ne considère que ceux qui sont présents dans l'échantillon. Étant données une collection de mots fréquents $\mathcal{W}_1$ dans un échantillon $\mathcal{S}_1$ et une collection de mots fréquents $\mathcal{W}_2$ dans un échantillon $\mathcal{S}_2$, on peut réaliser l'intersection de $\mathcal{W}_1$ et $\mathcal{W}_2$ pour obtenir une collection de mots fréquents dans $\mathcal{S}_1 \cup \mathcal{S}_2$. Étant donné un échantillon de n séquences, on peut le voir comme un ensemble de n échantillons et commencer par combiner les ensembles de mots fréquents jusqu'à ce que toutes les séquences soient combinées. Des approches particulières par séquençage diffèrent dans la façon de choisir les ensembles à combiner, ainsi que dans la manière de les combiner.

8.7 Îlots CG et le « casino équitable »

Le dinucléotide le plus rare dans de nombreux génomes est CG. En effet, le C de CG est typiquement méthylique et que le C-méthyle obtenu a tendance à muter en T. Cependant, la méthylation est supprimée autour des gènes situés dans les secteurs appelés des *îlots* CG, où CG apparaît relativement fréquemment. Comment alors définir et trouver des îlots CG dans un génome.

Trouver des îlots CG n'est pas très différent du problème du « jeu d'argent » suivant (Durbin *et al.*, 1998 [93]). Dans un « casino équitable », un donneur peut utiliser soit une pièce équilibrée, soit une pièce truquée dont la probabilité de faire face est $\frac{3}{4}$. Pour des raisons de sécurité, le donneur ne change pas les pièces — cela arrive relativement rarement, avec une probabilité de 0,1. Étant donnée une série de lancers, on doit découvrir quelle pièce a été utilisée par le donneur.

Résolvons d'abord le problème en sachant que le donneur ne change jamais de pièce. La question est de savoir quelle pièce, équilibrée ($p^+(0) = p^+(1) = \frac{1}{2}$) ou truquée ($p^-(0) = \frac{1}{4}, p^-(1) = \frac{3}{4}$), il a utilisé. Si la série de lancers obtenue est $x = x_1 \dots x_n$, alors la probabilité qu'elle ait été produite par une pièce équilibrée est $P(x|\textit{pièce équilibrée}) = \prod_{i=1}^{n} p^+(x_i) = \frac{1}{2^n}$. La probabilité que x provienne d'une pièce truquée est $P(x|\textit{pièce truquée}) = \prod_{i=1}^{n} p^-(x_i) = \frac{1}{4^{n-k}} \frac{3^k}{4^k} = \frac{3^k}{4^n}$, où k est le nombre de 1 présents dans x. Par suite, lorsque l'on a $k < \frac{n}{\log_2 3}$, il est plus probable que le donneur utilise une pièce équilibrée et, quand on a $k > \frac{n}{\log_2 3}$, il utilise plus sûrement une pièce truquée. On peut définir les quotients de chance logarithmiques comme suit :

$$\log_2 \frac{P(x|\textit{pièce équilibrée})}{P(x|\textit{pièce truquée})} = \sum_{i=1}^{k} \log_2 \frac{p^+(x_i)}{p^-(x_i)} = n - k \log_2 3.$$

Une approche naïve pour trouver les îlots CG consiste à calculer les quotients de chance logarithmiques pour chaque fenêtre coulissante de longueur fixée. Les fenêtres qui reçoivent des scores positifs sont des îlots CG potentiels. L'inconvénient d'une telle approche est que l'on ne connaît pas à l'avance la longueur des îlots CG. Les modèles de Markov cachés représentent une approche probabiliste différente de ce problème (Churchill, 1989 [69]).

8.8 Modèles de Markov cachés

Un *modèle de Markov caché* (HMM pour *Hidden Markov Model*) $\mathcal{M}$ est défini par un alphabet $\sum$, un ensemble d'états (cachés) Q, une matrice de probabilités de transition d'états A et une matrice de probabilités d'émission E, où

- $\sum$ est un alphabet de symboles,
- Q est un ensemble d'états qui émettent des symboles de l'alphabet $\sum$,
- $A = (a_{kl})$ est une matrice $|Q| \times |Q|$ de probabilités de transition d'états et
- $E = (e_k(b))$ est une matrice $|Q| \times |\sum|$ de probabilités d'émission.

Le lancer de pièces dans le « casino équitable » correspond au HMM suivant :

- $\sum = \{0, 1\}$, correspondant à pile (0) ou face (1) ;
- $Q = \{F, B\}$, correspondant à une pièce équilibrée ou truquée ;
- $a_{FF} = a_{BB} = 0,9$; $a_{FB} = a_{BF} = 0,1$;
- $e_F(0) = \frac{1}{2}$; $e_F(1) = \frac{1}{2}$; $e_B(0) = \frac{1}{4}$; $e_B(1) = \frac{3}{4}$.

Un chemin $\pi = \pi_1 \ldots \pi_n$ dans le HMM $\mathcal{M}$ est une suite d'états. Par exemple, si un donneur a utilisé la pièce équilibrée pour les trois premiers et les trois derniers lancers et la pièce truquée pour cinq lancers intermédiaires, le chemin correspondant est FFFBBBBBFFF. La probabilité qu'une séquence x soit produite par le chemin π (étant donné le modèle $\mathcal{M}$) est :

$$P(x|\pi) = \prod_{i=1}^{n} P(x_i|\pi_i)P(\pi_i|\pi_{i+1}) = a_{\pi_0,\pi_1} \times \prod_{i=1}^{n} e_{\pi_i}(x_i) \times a_{\pi_i,\pi_{i+1}}$$

où, par commodité, on introduit π_0 et π_{n+1} comme étant les états initiaux et terminaux fictifs *début* et fin.

Ce modèle définit la probabilité $P(x|\pi)$, pour une séquence x et un chemin π donnés. Cependant, seul le donneur connaît la vraie séquence d'états π qui a émis x. On dit alors que le chemin de x est caché ; il doit faire face au problème suivant :

Problème du décodage Trouver un chemin optimal $\pi^* = \arg\max_\pi P(x|\pi)$ pour x qui maximise $P(x|\pi)$.

La solution au problème du décodage est fournie par l'algorithme de Viterbi, 1967 [348], qui est une variation de l'approche de programmation dynamique de Bellman, 1957 [29]. L'idée est que le chemin optimal pour le $(i+1)$-préfixe $x_1 \ldots x_{i+1}$ de x utilise un chemin pour un i-préfixe de x qui est optimal parmi les chemins finissant dans un état (inconnu) $\pi_i = k \in Q$.

Par définition, $s_k(i)$ est la probabilité du chemin le plus probable pour le préfixe $x_1 \ldots x_i$ qui se termine avec l'état k ($k \in Q$ et $1 \leq i \leq n$). On a alors :

$$s_l(i+1) = e_l(x_{i+1}) \times \max_{k \in Q}\{s_k(i) \times a_{kl}\}.$$

On initialise $s_{début}(0) = 1$ et $s_k(0) = 0$ pour $k \neq$ *début*. La valeur de $P(x|\pi^*)$ est :

$$P(x|\pi^*) = \max_{k \in Q} s_k(n) a_{k,fin}.$$

L'algorithme de Viterbi fonctionne en un temps $O(n|Q|)$. Dans cet algorithme, les calculs se font habituellement avec des scores logarithmiques $S_l(i) = \log s_l(i)$ pour éviter tout débordement :

$$S_l(i+1) = \log e_l(x_{i+1}) + \max_{k \in Q}\{S_k(i) + \log(a_{kl})\}.$$

Étant donnée une suite de lancers x, quelle est la probabilité que le donneur possède une pièce truquée au moment i ? Une simple variante de l'algorithme de Viterbi nous permet de calculer la probabilité $P(\pi_i = k|x)$. Soit $f_k(i)$ la probabilité d'émettre le préfixe $x_1 \dots x_i$ et d'atteindre l'état $\pi_i = k$. On obtient alors :

$$f_k(i) = e_k(x_i) \times \sum_{l \in Q} f_l(i-1) \times a_{lk}.$$

La seule différence entre cet *algorithme* et celui de Viterbi est que le symbole « max » dans l'algorithme de Viterbi devient $\sum$ dans l'algorithme progressif. La probabilité rétrograde $b_k(i)$ est définie comme étant la probabilité d'être en l'état $\pi_i = k$ et d'émettre le suffixe $x_{i+1} \dots x_n$. L'*algorithme rétrograde* utilise une récurrence semblable :

$$b_k(i) = \sum_{l \in Q} e_l(x_{i+1}) \times b_l(i+1) \times a_{kl}.$$

Enfin, la probabilité que le donneur possède une pièce truquée au moment i est donnée par la formule suivante :

$$P(\pi_i = k|x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \cdot b_k(i)}{P(x)},$$

où $P(x, \pi_i = k)$ est la probabilité de x sachant que x_i a été produite en l'état k et $P(x) = \sum_\pi P(x|\pi)$.

8.9 Le casino d'Elkhorn et l'estimation des paramètres HMM

L'analyse précédente supposait que l'on connût les probabilités de transition et d'émission d'états du HMM. Le problème le plus difficile dans les applications des HMM est que ces paramètres sont inconnus et qu'il faut les estimer. Il est facile pour un joueur intelligent de deviner que le donneur du « casino équitable » utilise une pièce truquée. Il suffit de remarquer que 0 et 1 ont des fréquences moyennes différentes ($\frac{3}{8}$ et $\frac{5}{8}$, respectivement) et que, si l'on divise le nombre de 0 par le nombre de 1 lors d'une séquence de lancers d'une journée, on obtient un rapport étrangement faible. Cependant, il est beaucoup plus difficile d'estimer les probabilités de transition et d'émission du HMM correspondant. Dans « Belliou la fumée » de Jack London, Belliou fit l'une des premières tentatives pour imaginer les probabilités de transition d'une roulette dans le casino d'Elkhorn. Après avoir passé de longues heures et de longs jours à regarder la roulette, la nuit vint durant laquelle Belliou proclama qu'il était prêt à vaincre le système (n'essayez pas de le faire à Las Vegas, la technologie des jeux d'argent a changé).

Soit Θ un vecteur qui combine les probabilités de transition et d'émission inconnues du HMM $\mathcal{M}$. Étant donnée une chaîne x, on définit $P(x|\Theta)$ comme

étant la probabilité de x sachant les valeurs des paramètres Θ. Notre but est de trouver Θ^*, tel que l'on ait

$$\Theta^* = \arg\max_{\Theta} P(x|\Theta).$$

Habituellement, au lieu d'une seule chaîne x, on donne une collection de *séquences d'entraînement* $x^1, \ldots, x^m$ et le but est de maximiser

$$\Theta^* = \arg\max_{\Theta} \prod_{j=1}^{m} P(x^j|\Theta).$$

Ceci correspond à l'optimisation d'une fonction continue dans l'espace multidimensionnel des paramètres Θ. Les algorithmes couramment utilisés pour l'optimisation paramétrique sont des heuristiques fondées sur une stratégie d'amélioration locale dans l'espace des paramètres. Si le chemin $\pi_1 \ldots \pi_n$ correspondant aux états observés $x_1 \ldots x_n$ est connu, alors on peut explorer les séquences et calculer les estimations empiriques pour les probabilités de transition et d'émission. Si A_{kl} est le nombre de transitions de l'état k à l'état l et $E_k(b)$ le nombre de fois où b est émis par l'état k, alors les estimateurs du maximum de vraisemblance sont

$$a_{kl} = \frac{A_{kl}}{\sum_{q \in Q} A_{kq}} \quad \text{et} \quad e_k(b) = \frac{E_k(b)}{\sum_{\sigma \in \sum} E_k(\sigma)}.$$

Habituellement, la séquence d'états $\pi_1 \ldots \pi_n$ est inconnue et, dans ce cas, on utilise couramment une stratégie itérative d'amélioration locale appelée l'algorithme de *Baum-Welch* (Baldi et Brunak, 1997 [24]).

8.10 Alignement de profils HMM

Étant donnée une famille de séquences biologiques liées au niveau fonctionnel, on peut chercher de nouveaux membres de cette famille à l'aide d'alignements par paires entre ses membres et les séquences d'une base de données. Cependant, cette approche peut échouer dans la recherche de séquences d'une parenté éloignée. Une autre approche consiste à utiliser la totalité de l'ensemble des séquences liées fonctionnellement pour la recherche.

Un *profil* est la représentation la plus simple d'une famille de protéines apparentées qui est donnée par l'alignement multiple. Étant donné un alignement multiple de chaînes à n colonnes dans l'alphabet $\mathcal{A}$, un profil $\mathcal{P}$ est une matrice $|\mathcal{A}| \times n$ qui spécifie la fréquence $e_i(a)$ de chaque caractère a de l'alphabet $\mathcal{A}$ dans la colonne i (Gribskov *et al.*, 1987 [139]). Des profils peuvent être comparés et alignés les uns avec les autres, car l'algorithme de programmation dynamique qui permet d'aligner deux séquences fonctionne si les deux séquences de départ sont des alignements multiples (Waterman et Perlwitz, 1984 [362]).

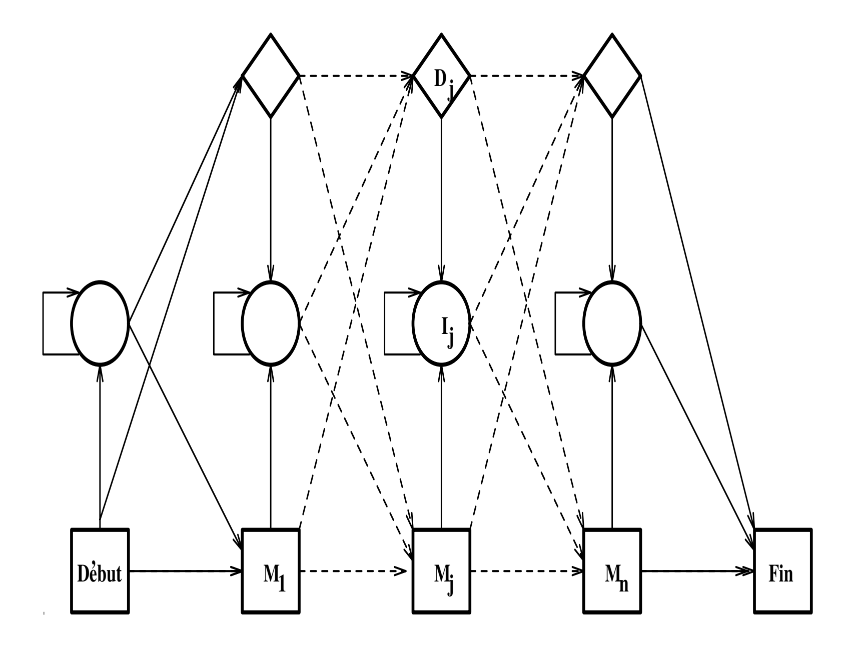


Figure 8.3 – Profil HMM.

Les HMM peuvent également être utilisés pour la comparaison de séquences (Krogh *et al.*, 1994 [208] et Sonnhammer *et al.*, 1997 [324]), plus particulièrement pour aligner une séquence et un profil. Le plus simple HMM pour un profil $\mathcal{P}$ contient n états d'*appariement* séquentiellement apparentés $M_1, \ldots, M_n$, de probabilités d'émission $e_i(a)$ prises à partir du profil P (figure 8.3). Étant donné le profil $\mathcal{P}$, la probabilité d'une chaîne $x_1 \ldots x_n$ est $\prod_{i=1}^{n} e_i(x_i)$. Pour modéliser les insertions et les délétions, on ajoute les états d'*insertions* $I_0, \ldots, I_n$ et les états de *délétions* $D_1, \ldots, D_n$ au HMM et on suppose :

$$e_{I_j}(a) = p(a),$$

où $p(a)$ est la fréquence du symbole a dans toutes les séquences. Les probabilités de transition entre les états d'appariement et d'insertion peuvent être définies dans le modèle de pénalité affine avec trous en fixant a_{MI}, a_{IM} et a_{II} de sorte que $\log(a_{MI}) + \log(a_{IM})$ soit égal à la pénalité de création de trous et que $\log(a_{II})$ soit égal à la pénalité d'extension de trous. Les états (silencieux) de délétion n'émettent pas de symbole.

On définit $v_j^M(i)$ comme étant le score de vraisemblance logarithmique du meilleur chemin pour apparier $x_1 \ldots x_i$ au profil HMM $\mathcal{P}$ qui finit par x_i et est émis par l'état M_j. On définit de façon semblable $v_j^I(i)$ et $v_j^D(i)$. Évidemment, la récurrence dynamique obtenue est très semblable à la récurrence d'alignement

standard s :

$$v_j^M(i) = \log \frac{e_{M_j}(x_i)}{p(x_i)} + \max \begin{cases} v_{j-1}^M(i-1) + \log(a_{M_{j-1},M_j}) \\ v_{j-1}^I(i-1) + \log(a_{I_{j-1},M_j}) \\ v_{j-1}^D(i-1) + \log(a_{D_{j-1},M_j}) \end{cases}$$

Les valeurs $v_j^I(i)$ et $v_j^D(i)$ sont définies de la même façon.

8.11 Échantillonnage de Gibbs

Lawrence *et al.*, 1993 [217] ont suggéré l'utilisation de l'*échantillonnage de Gibbs* pour trouver des modèles dans les séquences. Étant donnés un ensemble de séquences $x^1, \ldots, x^m$ dans un alphabet $\mathcal{A}$ et un entier w, le problème est de trouver une sous-chaîne de longueur w dans chaque x^i, de telle sorte que la similitude entre les m sous-chaînes soit maximale.

Soient $a^1, \ldots, a^m$ les indices de départ des sous-chaînes choisies dans $x^1, \ldots,$ x^m, respectivement. On note q_{ij} la fréquence avec laquelle le symbole i apparaît à la j-ième position des sous-chaînes.

L'échantillonnage de Gibbs est une procédure itérative qui, à chaque itération, met de côté une séquence de l'alignement et la remplace par une nouvelle. L'échantillonnage de Gibbs débute en choisissant aléatoirement des sous-chaînes de longueur w dans chacune des m séquences $x^1, \ldots, x^m$ et procède comme suit :

– Au début de chaque itération, on choisit une sous-chaîne de longueur w dans chacune des m séquences $x^1, \ldots, x^m$.

– On choisit aléatoirement l'une des séquences x^r, toutes les séquences étant équiprobables.

– On crée une matrice des fréquences (q_{ij}) à partir des $m - 1$ sous-chaînes restantes.

– Pour chaque position i dans x^r, on calcule la probabilité $p_i = \prod_{j=0}^{w} q_{x^r_{i+j},j}$ que la sous-chaîne qui commence en cette position soit produite par le profil (q_{ij}) (x^r_{i+j} désigne le symbole situé en position $i + j$ dans la séquence x^r).

– On choisit la position de départ i de la nouvelle sous-chaîne dans x^r au hasard, avec une probabilité proportionnelle à p_i.

Bien que l'échantillonnage de Gibbs soit connu pour fonctionner dans des cas spécifiques, il peut, comme l'algorithme de Baum-Welch, converger vers un maximum local. La procédure décrite (Lawrence *et al.*, [217]) ne permettant

pas les insertions et les délétions, Rocke et Tompa, 1998 [288] ont généralisé cette méthode pour la prise en compte des trous dans un modèle.

8.12 Quelques autres problèmes et approches

8.12.1 Trouver des signaux avec trous

Rigoutsos et Floratos, 1998 [285] se sont intéressés au problème qui consiste à trouver des signaux avec trous dans un texte. Par définition, une chaîne avec trous est une chaîne constituée d'une combinaison arbitraire de symboles de l'alphabet et de symboles « joker ». Une chaîne P avec trous est appelée une $< l, w >$-chaîne si toute sous-chaîne de P de longueur l contient au moins w symboles de l'alphabet. L'algorithme TERESIAS (Rigoutsos et Floratos, 1998 [285]) trouve tous les $< l, w >$-modèles *maximaux* qui apparaissent dans au moins K séquences de l'échantillon, avec une approche à deux étages. Au premier étage d'*exploration*, il trouve toutes les courtes chaînes de longueur l avec au moins w symboles de l'alphabet qui apparaissent au moins K fois dans l'échantillon. Au second étage de *circonvolution*, il assemble ces courtes chaînes en des (l, w)-chaînes maximales.

8.12.2 Trouver des signaux dans des échantillons avec des fréquences truquées

Le problème du mot magique devient difficile si le signal n'est contenu que dans une fraction de toutes les séquences et si la distribution nucléotidique de fond dans l'échantillon est biaisée. Dans ce cas, chercher un signal ayant le nombre maximal d'occurrences peut aboutir à des modèles composés des nucléotides les plus fréquents. Ces modèles peuvent ne pas être importants biologiquement. Par exemple, si A a une fréquence de 70% et que T, G et C ont des fréquences de 10%, alors poly(A) peut être le mot le plus fréquent, déguisant ainsi le vrai mot magique.

Pour trouver des mots magiques dans des échantillons truqués, de nombreux algorithmes utilisent l'*entropie relative* pour mettre le mot magique au premier plan, parmi les mots composés de nucléotides fréquents. Étant donné un mot magique de longueur k, l'entropie relative est définie de la façon suivante :

$$\sum_{j=1}^{k} \sum_{r=A,T,G,C} p_{rj} \log_2 \frac{p_{rj}}{b_r},$$

où p_{rj} est la fréquence du nucléotide r en position j parmi les occurrences du mot magique et b_r la fréquence de fond de r.

L'entropie relative est une bonne mesure pour comparer deux mots magiques ayant le même nombre d'occurrences dans l'échantillon, mais ce n'est pas une bonne mesure si les mots apparaissent dans des nombres de séquences

très différents. Hertz et Stormo, 1999 [159] et Tompa, 1999 [338] se sont intéressés à ce problème en créant un critère qui rend compte à la fois du nombre d'occurrences et de la distribution du bruit de fond.

8.12.3 Choix de l'alphabet dans la découverte de signaux

Karlin et Ghandour, 1985 [187] ont observé qu'il n'y a aucun moyen de savoir à l'avance quel alphabet choisir pour révéler des signaux dans l'ADN ou les protéines. Par exemple, $WSWS \ldots WSWS$ peut être un signal très fort ; il est difficile à trouver dans l'alphabet standard A, T, G, C. Ce problème a été étudié par Sagot *et al.*, 1997 [293].

Chapitre 9

Prédiction génétique

9.1 Introduction

Dans les années 60, Charles Yanofsky, Sydney Brenner et leurs collaborateurs ont montré qu'un gène et son produit protéique sont des structures colinéaires avec une corrélation directe entre les triplets de nucléotides du gène et les acides aminés de la protéine. Cependant, le concept de gène vu comme une chaîne synthétique de nucléotides n'a pas survécu longtemps. Les gènes se chevauchant et les gènes à l'intérieur des gènes ont été découverts à la fin des années 60. Ces études ont démontré que le problème calculatoire de la prédiction génétique était loin d'être simple. Enfin, en 1977, la découverte de gènes humains scindés a créé une énigme calculatoire de prédiction génétique.

Les génomes eucaryotes sont plus grands et plus complexes que les génomes procaryotes. Ceci n'est pas vraiment surprenant, puisque l'on peut espérer trouver davantage de gènes chez les humains que chez les bactéries. Cependant, la taille du génome de nombreux eucaryotes ne semble pas liée à la complexité génétique ; par exemple, le génome de la salamandre est dix fois plus grand que celui de l'être humain. Les eucaryotes ne contiennent pas seulement des gènes mais aussi de grandes quantités d'ADN qui ne codent aucune protéine (ADN « poubelle »). En outre, la plupart des gènes humains sont interrompus par de l'ADN poubelle et cassés en morceaux appelés des exons. La différence de taille entre les génomes de la salamandre et de l'être humain tient donc à une quantité d'ADN poubelle et de répétitions plus grande chez la salamandre que chez l'homme.

Les gènes fendus ont été découverts en 1977 de façon indépendante par les laboratoires de Phillip Sharp et Richard Roberts, lors d'études sur l'adénovirus (Berget *et al.*, 1977 [32] et Chow *et al.*, 1977 [67]). Cette découverte a été une surprise telle que l'article du groupe de Richard Roberts avait un titre étonnant pour le magazine très académique *Cell* : « Un arrangement de séquences stupéfiant à l'extrémité 5' de l'adénovirus ARNm de type 2 ». Berget *et al.*, 1977 [32] ont concentré leurs expériences sur un ARNm qui code

une protéine virale connue comme exon. Pour cartographier l'ARNm exon sur le génome viral, l'ARNm a été hybridé à l'ADN adénovirus et les molécules hybrides ont été analysées par microscopie électronique. De manière frappante, les hybrides ARNm-ADN formés dans cette expérience ont exposé trois structures de boucles au lieu du double segment continu suggéré par le modèle classique du « gène continu ». D'autres expériences d'hybridation ont révélé que l'ARNm exon est construit à partir de quatre fragments séparés du génome de l'adénovirus. Ces quatre exons dans le génome de l'adénovirus sont séparés par trois fragments « poubelle » appelés des *introns*. La découverte des gènes fendus (*épissage*) dans l'adénovirus a rapidement été suivi par l'évidence selon laquelle les gènes des mammifères avaient également des structures fendues (Tilghman *et al.*, 1978 [337]). Ces études expérimentales ont fait naître un problème calculatoire de prédiction génétique qui est encore non résolu : les gènes humains ne représentent que 3% du génome humain et aucun algorithme connu de reconnaissance de gènes *in silico* ne fournit une reconnaissance génétique fiable.

Après avoir séquencé un nouveau fragment d'ADN, les biologistes tentent d'y trouver des gènes. La manière statistique traditionnelle d'attaquer ce problème a consisté à chercher des caractéristiques qui apparaissent fréquemment dans les gènes et rarement ailleurs. De nombreux chercheurs ont utilisé une approche plus biologique et ont essayé de reconnaître les localisations des signaux d'épissage au niveau des jonctions exon-intron. Le but d'une telle approche est la caractérisation des sites de l'ARN, où les protéines et les ribonucléoprotéines impliquées dans le mécanisme d'épissage se lient/interagissent. Par exemple, les dinucléotides AG et GT sur les côtés gauche et droit des exons sont hautement conservés. La façon la plus simple de représenter un signal est de donner un modèle consensus constitué du nucléotide le plus fréquent à chaque position d'un alignement de signaux spécifiques. Bien que des catalogues de sites épissés aient été compilés au début des années 80, les modèles consensus ne sont pas très fiables pour faire la distinction entre de vrais sites et des pseudo-sites, car ils ne contiennent pas d'information sur les fréquences nucléotidiques à différentes positions. Ed Trifonov a inventé un exemple qui montre un autre défaut potentiel des consensus :

MELON
MANGO
HONEY
SWEET
COOKY
——
MONEY

L'information de fréquence est captée par les *profils* (ou *matrices scores-positions*), qui associent des scores basés sur les fréquences à chaque nucléotide possible en chaque position du signal. Malheureusement, l'utilisation de profils pour la prédiction de sites d'épissage a eu un succès limité, certainement à

cause de la coopération entre les molécules de liaisons multiples. Les tentatives pour améliorer la précision de la prédiction génétique ont mené à l'utilisation de réseaux neuronaux et de modèles de Markov cachés pour la découverte de gènes.

Les projets de séquençage à grande échelle ont motivé le besoin d'une nouvelle génération d'algorithmes pour la reconnaissance génétique. La prédiction génétique par similitude se fonde sur l'observation selon laquelle un gène nouvellement séquencé possède une bonne chance d'avoir une relation déjà connue dans la base de données (Bork et Gibson, 1996 [41]). Le flux de nouvelles données de séquençage va encore accroître cette chance. Par suite, la tendance en prédiction génétique à la fin des années 90 est passée des approches reposant sur les statistiques à celles qui se fondent sur les similitudes et sur EST. En particulier, Gelfand *et al.*, 1996 [125] ont proposé une approche combinatoire à la prédiction génétique, qui utilise des protéines voisines pour en déduire la structure exon-intron. Au lieu de recourir aux propriétés statistiques des exons, cette méthode tente de résoudre une énigme combinatoire : trouver un ensemble de sous-chaînes dans une séquence génomique dont la concaténation (épissage) s'ajuste aux protéines connues.

Après avoir réalisé leurs prédictions, les biologistes tentent de les vérifier expérimentalement. Cette vérification revient habituellement à un séquençage de l'ARNm sur toute sa longueur. Comme ce procédé a une complexité temporelle assez élevée, les prédictions *in silico* font leur chemin dans les bases de données et aboutissent fréquemment à des erreurs d'annotation. On ne peut que deviner la quantité de séquences annotées de façon incorrecte dans GenBank, mais il est clair que le nombre de gènes ayant été annotés sans les données concernant l'ARNm dans toute sa longueur (qui sont donc potentiellement erronés) peut être grand. Le problème du développement d'un algorithme de prédiction génétique « prêt à annoter » et celui de la correction de ces erreurs demeurent ouverts.

9.2 Approche statistique pour la prédiction génétique

La façon la plus simple de détecter des régions codantes potentielles est de regarder les *cadres de lecture ouverte* (ORF pour *Open Reading Frames*). Une ORF est une séquence de codons dans l'ADN qui commence par un codon start, se termine par un codon stop et ne possède aucun autre codon stop à l'intérieur. On espère trouver des codons stops fréquents dans de l'ADN non codant, simplement parce que trois des soixante quatre codons possibles sont des terminateurs de translation. La distance moyenne entre les codons stops dans de l'ADN « aléatoire » est de $\frac{64}{3} \approx 21$, nettement inférieure au nombre de codons dans une protéine moyenne (environ 300). Ainsi, de longues ORF signalent des gènes potentiels (Fickett, 1996 [105]), bien qu'elles ne parviennent pas à détecter des gènes courts ou des gènes avec de courts exons.

De nombreux algorithmes de prédiction génétique reposent sur la reconnaissance de régularités diffuses dans les régions protéiques codantes, comme le biais dans l'*usage de codon* (ou *biais de codon*). L'usage de codon est un vecteur 64-mère qui donne les fréquences de chacun des 64 *codons* possibles (triplets de nucléotides) dans une fenêtre. Les vecteurs d'usage de codon diffèrent entre les fenêtres codantes et non codantes, permettant ainsi d'utiliser cette mesure pour la prédiction génétique (Fickett, 1982 [104] et Staden et McLachlan, 1982 [327]). Gribskov *et al.*, 1984 [138] ont utilisé une approche avec un rapport de vraisemblance pour calculer les probabilités conditionnelles de la séquence d'ADN dans une fenêtre, sous l'hypothèse d'une séquence aléatoire codante et sous celle d'une non codante. Lorsque la fenêtre glisse le long de l'ADN, les gènes sont souvent révélés par des pics sur les représentations graphiques du rapport de vraisemblance. Le *dénombrement d'hexamères en phase* est un meilleur détecteur codant ; il est semblable aux modèles de Markov d'ordre trois cinquièmes (Borodovsky et McIninch, 1993 [42]). Fickett et Tung, 1992 [106] ont évalué de nombreuses mesures codantes de ce genre et sont parvenus à la conclusion qu'elles donnent une image de résolution plutôt faible des bords des régions codantes, avec de nombreuses associations de faux positifs et de faux négatifs. En outre, la structure exon-intron complique l'application de ces techniques aux eucaryotes. La longueur moyenne des exons chez les vertébrés est de 130 bp ; les exons sont donc souvent trop courts pour produire des pics dans le graphe de la fenêtre glissante.

L'usage de codon, l'usage d'acide aminé, les périodicités dans les régions codantes et autres paramètres statistiques (voir Gelfand, 1995 [123] pour un compte rendu) n'ont probablement rien en commun avec la façon dont le mécanisme d'épissage reconnaît les exons. De nombreux chercheurs ont utilisé une approche davantage inspirée de la biologie et ont tenté de reconnaître les localisations des signaux d'épissage aux jonctions exon-intron (Brunak *et al.*, 1991 [50]). Il existe une séquence (faiblement) conservée de huit nucléotides au bord d'un exon et d'un intron (site d'épissage *donneur*) et une séquence de quatre nucléotides au bord de l'intron et de l'exon (site d'épissage *accepteur*). Malheureusement, les profils pour la prédiction de sites d'épissages ont eu un succès limité, certainement à cause de la coopération entre des molécules agglomérantes multiples. Les profils sont équivalents à un type simple de réseau neuronal appelé perceptron. Des réseaux neuronaux plus compliqués (Uberbacher et Mural, 1991 [339]) et les modèles de Markov cachés (Krogh *et al.*, 1994 [209] et Burge et Karlin, 1997 [54]) captent les dépendances statistiques entre les sites et améliorent la qualité des prédictions.

De nombreux chercheurs ont tenté de combiner régions codantes et prédictions de signaux d'épissage en une construction de signaux. Par exemple, une prédiction de site d'épissage est plus crédible si les signes d'une région codante apparaissent d'un côté du site et pas de l'autre. En raison des limites des statistiques individuelles, quelques groupes ont développé des algorithmes de prédiction génétique qui combinent de multiples indices en une simple construction (Nakata *et al.*, 1985 [249], Gelfand, 1990 [121], Guigo *et al.*, 1992 [142] et

Snyder et Stormo, 1993 [321]). La construction du modèle de Markov caché de GENSCAN utilise quasiment toutes les statistiques existantes (Burge et Karlin, 1997 [54]). Cet algorithme ne se contente pas de fusionner les statistiques du site d'épissage, du promoteur, du site de polyadénylation et de la région codante, il prend également en compte leur non-homogénéité. Ceci a permis aux auteurs de dépasser les 90% d'exactitude dans les prédictions génétiques statistiques. Cependant, l'exactitude décroît de façon significative pour les gènes qui possèdent de nombreux exons courts ou pour ceux qui ont un usage de codon inhabituel.

9.3 Approche fondée sur la similitude pour la prédiction génétique

La similitude pour la détection génétique a d'abord été proposée par Gish et States, 1993 [129]. Bien que la recherche en similitude ait longtemps été utilisée pour la *détection* génétique (c'est-à-dire répondre à la question de savoir si un gène est présent dans un fragment d'ADN donné), le potentiel de la recherche en similitude pour la *prédiction* génétique (c'est-à-dire non seulement pour la détection mais aussi pour la prédiction détaillée de la structure exon-intron) a été peu exploité jusqu'au milieu des années 90. Snyder et Stormo, 1995 [322] ainsi que Searls et Murphy, 1995 [313] ont fait les premiers essais pour incorporer l'analyse en similitude dans les algorithmes de prédiction génétique. Cependant, la complexité calculatoire de l'exploration de tous les assemblages d'exons au début des algorithmes d'alignement de séquences est plutôt élevée.

Gelfand *et al.*, 1996 [125] ont proposé une approche reposant sur l'alignement épissé pour le problème de l'assemblage d'exons ; celle-ci utilise des protéines voisines pour trouver la structure exon-intron. La figure 9.1a illustre le problème de l'alignement épissé pour la séquence « génomique » suivante :

It was brilliant thrilling morning and the slimy hellish lithe doves

gyrated and gambled nimbly in the waves,

dont les différents blocs composent la fameuse ligne de Lewis Carroll :

't was brillig, and the slithy toves did gyre and gimble in the wabe

L'approche de Gelfand *et al.*, 1996 [125] repose sur l'idée suivante (illustrée par Oksana Khleborodova). Étant donnée une séquence génomique (voir figure 9.2), on trouve d'abord un ensemble de *blocs candidats* contenant tous les exons *exacts* (voir figure 9.3). Ceci s'obtient en sélectionnant tous les blocs situés entre les sites *accepteurs* et *donneurs* potentiels (c'est-à-dire entre les dinucléotides AG et GT), avec une séparation ultérieure par *filtrage* de cet ensemble (de façon à ne pas perdre les véritables exons). Évidemment, l'ensemble des blocs obtenu peut contenir de nombreux faux exons et, généralement, il

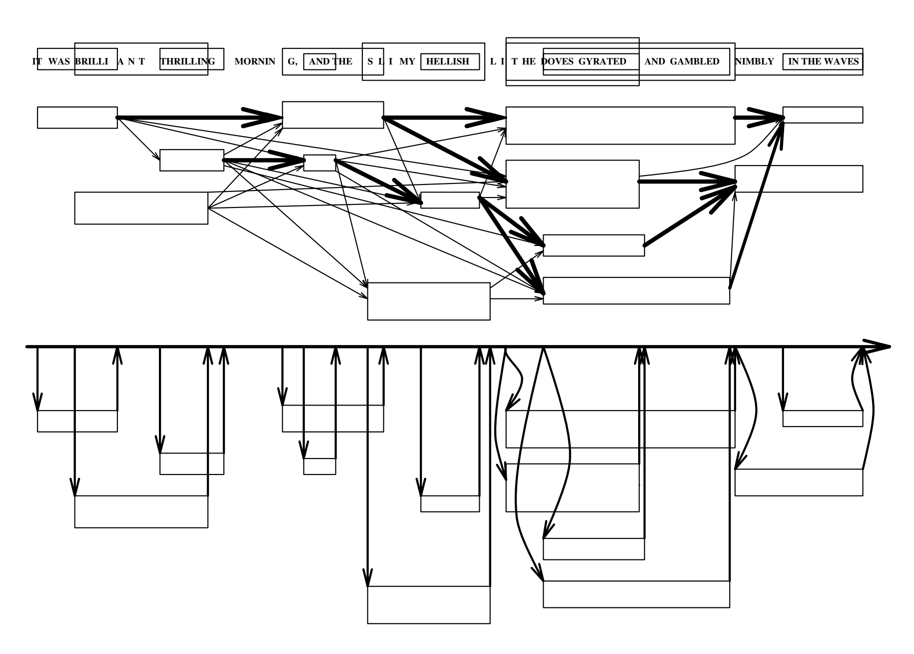


Figure 9.1 – Problème de l'alignement épissé : a) assemblages des blocs qui s'ajustent le mieux à la ligne de Lewis Carroll, b) réseau d'alignement correspondant et c) transformation équivalente du réseau d'alignement.

est impossible de distinguer tous les exons véritables de cet ensemble par une procédure statistique. Plutôt que d'essayer de trouver les exons réels, Gelfand *et al.*, 1996 [125] sélectionnent une protéine *cible* voisine dans GenBank (voir figure 9.4) et explorent tous les assemblages de blocs possibles, dans le but de trouver un assemblage dont le score de similitude avec la protéine cible est le plus élevé (voir figure 9.5). Le nombre d'assemblages de blocs différents est très grand (voir figures 9.6, 9.7 et 9.8), mais l'algorithme d'*alignement épissé*, ingrédient clé de cette méthode, les explore tous en un temps polynomial (voir figure 9.9).

9.4 Alignement épissé

Soit $G = g_1 \dots g_n$ une chaîne et soient $B = g_i \dots g_j$ et $B' = g_{i'} \dots g_{j'}$ des sous-chaînes de G. On écrit $B \prec B'$ si ces sous-chaînes vérifient $j < i'$, autre-



Figure 9.2 – Étude d'une séquence génomique.

ment dit si B se termine avant le début de B'. Une séquence $\Gamma = (B_1, \ldots, B_p)$ de sous-chaînes de G est une *chaîne* si elle vérifie $B_1 \prec B_2 \prec \cdots \prec B_p$. On note $\Gamma^* = B_1 * B_2 * \ldots * B_p$ la *concaténation* des chaînes de la chaîne Γ. Étant données deux chaînes G et T, $s(G, T)$ désigne le score de l'*alignement optimal* entre G et T.

Soient $G = g_1 \ldots g_n$ une chaîne appelée *séquence génomique*, $T = t_1 \ldots t_m$ une chaîne appelée *séquence cible* et $\mathcal{B} = \{B_1, \ldots B_b\}$ un ensemble de sous-chaînes de G que l'on appelle des *blocs*. Étant donnés G, T et $\mathcal{B}$, le *problème de l'alignement épissé* consiste à trouver une chaîne Γ de chaînes de $\mathcal{B}$ telle que le score d'alignement $s(\Gamma^*, T)$ entre la concaténation de ces chaînes et la séquence cible soit maximal parmi toutes les chaînes de blocs de $\mathcal{B}$.

Une approche naïve au problème de l'alignement épissé consiste à détecter toutes les similitudes relativement grandes entre chaque bloc et la séquence cible, puis à les assembler en un sous-ensemble optimal de fragments semblables



Figure 9.3 – Séparation par filtrage des exons candidats.

compatibles. L'inconvénient de cette approche est que le nombre de blocs est généralement très élevé et que les extrémités des domaines de similitude ne sont pas bien définies.

Gelfand *et al.*, 1996 [125] ont réduit le problème de l'assemblage d'exons à la recherche d'un chemin dans un graphe orienté (voir figure 9.1b) ; ses sommets correspondent aux blocs et ses arêtes aux transitions potentielles entre les blocs. Par définition, le poids d'un chemin est le poids de l'alignement optimal entre les blocs concaténés de ce chemin et la séquence cible. Notons que le problème de l'assemblage d'exons est différent du problème standard du chemin minimal (les poids des sommets et des arêtes dans le graphe ne sont même pas définis).

Soit $B_k = g_m \dots g_i \dots g_l$ une sous-chaîne de G contenant une position i. Par définition, le *i-préfixe* de B_k est $B_k(i) = g_m \dots g_i$. Pour un bloc $B_k = g_m \dots g_l$, on pose $prem(k) = m$, $dern(k) = l$ et $taille(k) = l - m + 1$. Soit $\mathcal{B}(i) = \{k : \quad dern(k) < i\}$ l'ensemble des blocs se terminant (strictement)



Figure 9.4 – Découverte d'une protéine cible.

avant la position i dans G. Soit $\Gamma = (B_1, \ldots, B_k, \ldots, B_t)$ une chaîne telle qu'un certain bloc B_k contienne la position i. On définit $\Gamma^*(i)$ comme étant une chaîne $\Gamma^*(i) = B_1 * B_2 * \ldots * B_k(i)$. Soit :

$$S(i, j, k) = \max_{\text{toutes les chaînes } \Gamma \text{ contenant le bloc } B_k} s(\Gamma^*(i), T(j)).$$

La récurrence suivante calcule $S(i, j, k)$ pour $1 \leq i \leq n$, $1 \leq j \leq m$ et $1 \leq k \leq b$. Pour simplifier, on considère l'alignement de séquences avec des pénalités de trous *linéaires* et on définit $\delta(x, y)$ comme étant le score de similitude pour toute paire d'acides aminés x et y et δ_{indel} comme une pénalité pour l'insertion ou la délétion d'acides aminés.



Figure 9.5 – Utilisation de la protéine cible comme patron pour l'assemblage d'exons.

$$S(i,j,k) =$$

$$\max \begin{cases} S(i-1,j-1,k) + \delta(g_i,t_j), & \text{si } i \neq prem(k) \\ S(i-1,j,k) + \delta_{indel}, & \text{si } i \neq prem(k) \\ \max_{l \in \mathcal{B}(prem(k))} S(dern(l),j-1,l) + \delta(g_i,t_j), & \text{si } i = prem(k) \\ \max_{l \in \mathcal{B}(prem(k))} S(dern(l),j,l) + \delta_{indel}, & \text{si } i = prem(k) \\ S(i,j-1,k) + \delta_{indel} & \end{cases} \quad (9.1)$$

Une fois calculé le tableau S de dimension 3, le score de l'alignement épissé optimal est donné par :

$$\max_k S(dern(k), m, k).$$

Le problème de l'alignement épissé peut également être formulé comme un problème d'alignement de *réseaux* (Kruskal et Sankoff, 1983 [211]). Dans cette



Figure 9.6 – Assemblage.

formulation, chaque bloc B_k correspond à un chemin de longueur $taille(k)$ entre les sommets $prem(k)$ et $dern(k)$ et les chemins correspondant aux blocs B_k et B_t sont reliés par une arête $(dern(k), prem(t))$ s'ils vérifient $B_k \prec B_t$ (voir figure 9.1b). Le problème d'alignement de réseaux consiste à trouver un chemin dans le réseau qui présente le meilleur alignement avec la séquence cible.

Gelfand *et al.*, 1996 [125] ont réduit le nombre d'arêtes dans le graphe d'alignement épissé en faisant des transformations équivalentes du réseau décrit, aboutissant ainsi à une réduction de temps et d'espace. On définit :

$$P(i, j) = \max_{l \in \mathcal{B}(i)} S(dern(l), j, l).$$



Figure 9.7 – Et assemblage...

L'égalité (9.1) peut donc être réécrite en :

$$S(i,j,k) = \max \begin{cases} S(i-1,j-1,k) + \delta(g_i,t_j), & \text{si } i \neq prem(k) \\ S(i-1,j,k) + \delta_{indel}, & \text{si } i \neq prem(k) \\ P(prem(k),j-1) + \delta(g_i,t_j), & \text{si } i = prem(k) \\ P(prem(k),j) + \delta_{indel}, & \text{si } i = prem(k) \\ S(i,j-1,k) + \delta_{indel} & \end{cases} \quad (9.2)$$

où :

$$P(i,j) = \max \begin{cases} P(i-1,j) \\ \max_{k:\ dern(k)=i-1} S(i-1,j,k) \end{cases} \quad (9.3)$$

Le réseau correspondant à (9.2) et (9.3) possède un nombre d'arêtes sensiblement inférieur à celui de (9.1) (voir figure 9.1c), aboutissant donc à une implémentation pratique de l'algorithme d'alignement épissé.



Figure 9.8 – Et assemblage...

L'approche la plus simple pour la construction de blocs $\mathcal{B}$ est de produire tous les fragments entre les sites d'épissage potentiels représentés par AG (site accepteur) et GT (site donneur), à l'exception des blocs ayant des codons stops dans les trois structures. Cependant, cette approche crée un problème, car elle produit de nombreux blocs courts. Des expériences sur l'algorithme d'alignement épissé ont révélé que des prédictions incorrectes pour des cibles distantes sont fréquemment associées à l'*effet mosaïque* causé par des exons potentiels très courts. Le problème est que ces exons courts peuvent être facilement combinés pour s'ajuster à toute protéine cible. Il est plus facile de « composer » une phrase donnée à partir d'un millier de chaînes aléatoires si elles sont courtes. Par exemple, la phrase « filtrage d'exons candidats » peut très probablement être composée à partir d'un échantillon d'un millier de chaînes aléatoires de deux lettres (« fi », « lt », « ra », etc. ont des chances d'être présentes dans l'échantillon). La probabilité que la même phrase puisse être composée à par-

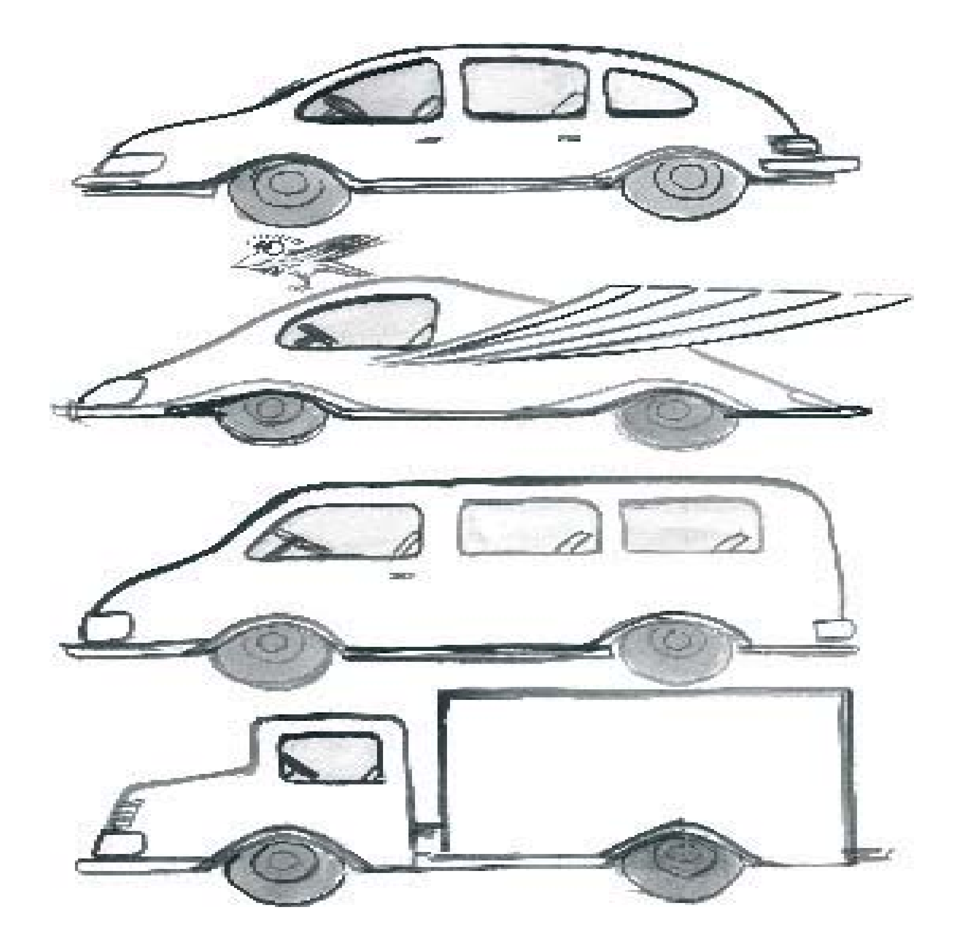


Figure 9.9 – Sélection du meilleur assemblage d'exons.

tir d'un échantillon du même nombre de chaînes aléatoires de cinq lettres est proche de zéro (le simple fait de trouver la chaîne « filtr » dans cet échantillon est peu probable). Cette observation explique l'effet mosaïque : si le nombre de blocs courts est élevé, les chaînes de ces blocs peuvent remplacer les véritables exons dans les alignements épissés, aboutissant ainsi à des prédictions avec un nombre anormalement grand d'exons courts. Pour éviter l'effet mosaïque, les exons candidats sont soumis à une procédure de (faible) filtrage ; par exemple, seuls les exons ayant un fort potentiel codant sont retenus.

Après avoir trouvé l'assemblage de blocs optimal, on espère qu'il représente la structure exon-intron correcte. Ceci est presque garanti si une protéine suffisamment semblable à celle codée dans le fragment analysé est réalisable : 99% de corrélation avec les gènes réels peuvent être obtenus à partir des cibles situées à une distance inférieure ou égale à 100 PAM (40% de similitude). L'algorithme d'alignement épissé fournit des prédictions très précises si même

une protéine de parenté éloignée est réalisable : les prédictions de 160 PAM (25% de similitude) sont encore fiables (95% de corrélation). Si l'on connaît une protéine de mammifère voisine d'un gène humain analysé, l'exactitude des prédictions génétiques dans ce fragment est d'environ 97% à 99% ; elle est de 95%, 93% ou 91% pour des protéines voisines de plantes, de champignons ou de procaryotes, respectivement (Mironov *et al.*, 1998 [242]). Un progrès supplémentaire a été obtenu en prédiction génétique en utilisant les données EST pour la prédiction génétique par similitude. En particulier, en utilisant les assemblages EST, Mironov *et al.*, 1999 [240] ont trouvé un grand nombre de gènes épissés différemment.

9.5 Découverte de gènes inversée et localisation d'exons dans l'ADNc

La découverte de gènes suit souvent le paradigme de *clonage positionnel* qui suit :

séquençage de l'ADN génomique → structure exons → ARNm → protéine

Dans les projets de clonage positionnel, les séquences génomiques sont les sources d'information fondamentales pour la prédiction génétique, la détection de mutations et la recherche avancée de gènes responsables de maladies. Le passage du clonage positionnel au paradigme de la *banque de gènes candidats* consiste à inverser le chemin traditionnel de la découverte de gènes en celui-ci :

protéine/ARNm → structure exons → séquençage de l'ADN génomique limité

Ainsi, les travaux récents pour la découverte de gènes passent du clonage positionnel d'un seul gène à l'analyse de maladies polygéniques avec des banques de gènes candidats composées de centaines de gènes. Les gènes qui forment une banque de gènes candidats peuvent provenir de différentes sources, comme par exemple l'analyse d'expression, le criblage d'anticorps, la protéomique, etc. Évidemment, des centaines de tentatives de clonage positionnel sont trop coûteuses pour être pratiquées.

Pour trouver le gène responsable d'une maladie, le clonage positionnel commence par la cartographie génétique ; on procède à la détection des mutations liées à la maladie. On a besoin d'une multitude d'étapes pour inclure le clonage génomique de grands fragments d'ADN, le criblage de banques d'ADNc, l'isolement de l'ADNc, le sous-clonage de grands clones génomiques pour le séquençage, etc. (voir figure 9.10). Dans de nombreux travaux de recherche génétique, la motivation majeure pour les étapes de sous-clonage génomique et de séquençage réside dans la détermination des bords exon-intron des gènes. Cette étape est souvent critique. Elle nécessite la fabrication d'amorces PCR introniques encadrant chaque exon. Traditionnellement, les bords des exons sont obtenus en comparant l'ADNc et les séquences génomiques. Le procédé

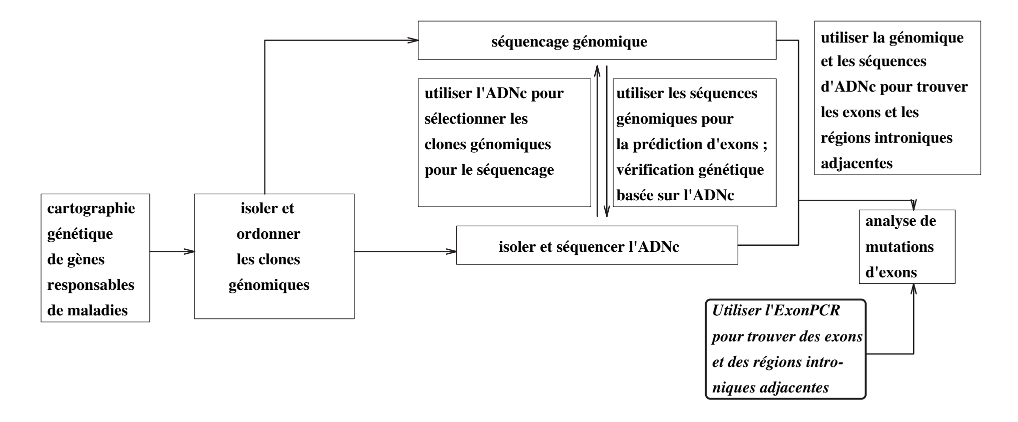


Figure 9.10 – Le clonage positionnel et l'ExonPCR.

tout entier peut être très coûteux en temps et impliquer de multiples étapes de sous-clonage, ainsi qu'un vaste séquençage.

L'ExonPCR (Xu *et al.*, 1998 [372]) est un protocole expérimental alternatif qui explore le chemin « inverse » de la découverte de gènes et fournit une transition rapide entre la découverte de l'ADNc et la détection de mutations (voir figure 9.10). L'ExonPCR trouve les bords des exons « cachés » dans l'ADNc (plutôt que dans l'ADN génomique) et ne nécessite pas le séquençage de clones génomiques. Durant la première étape, l'ExonPCR localise les positions approximatives des bords des exons dans l'ADNc par la PCR sur l'ADN génomique, à l'aide de sondes créées à partir de la séquence d'ADNc. La seconde étape consiste à effectuer la LM-PCR (pour *ligation-mediated PCR*) pour trouver les régions introniques situées sur les flancs. Par conséquent, l'effort déployé pour séquencer de l'ADN peut être considérablement réduit.

Les approches calculatoires pour la découverte des bords des exons dans l'ADNc (Gelfand, 1992 [122] et Solovyev *et al.*, 1994 [323]) ont exploré les *ombres d'épissage* (c'est-à-dire des parties des signaux d'épissage présents dans les exons). Cependant, comme les signaux des ombres d'épissage sont très faibles, les prédictions correspondantes ne sont pas fiables. L'ExonPCR est une approche expérimentale qui permet de trouver les bords des exons dans l'ADNc ; elle utilise les amorces PCR dans une série d'itérations. Les amorces sont créées à partir de la séquence d'ADNc et sont utilisées pour amplifier l'ADN génomique. Chaque paire d'amorces fait office de requête qui demande si, dans l'ADN génomique, il existe un intron ou des introns entre les séquences amorces. On obtient la réponse à cette question en comparant la longueur des produits de la PCR dans l'ADNc et l'ADN génomique. Si ces longueurs coïncident, les amorces appartiennent au même exon ; sinon, il existe un bord d'exon entre les amorces correspondantes. Chaque paire d'amorces donne une réponse oui/non, sans révéler les positions exactes des bords des exons. Le but est de les trouver et de minimiser à la fois le nombre d'amorces et le nombre d'itérations.

Différents types de stratégies peuvent être utilisés et le problème est semblable au jeu des « vingt questions » avec des gènes. La différence avec le jeu de société est que les gènes ont une option « pas de réponse » et que, parfois, ils peuvent donner une réponse fausse et restreindre les types de requêtes possibles. Ceci est semblable au « jeu des vingt questions avec un menteur » (Lawler et Sarkissian, 1995 [216]), mais ceci implique de nombreuses contraintes supplémentaires, y compris des bornes inférieure et supérieure à la longueur des requêtes (distance entre les amorces PCR).

L'ExonPCR tente d'imaginer une stratégie qui minimise le nombre total d'amorces PCR (pour réduire le coût) et qui, dans le même temps, minimise le nombre d'itérations requises pour les expériences PCR (pour réduire le temps). Cependant, ces objectifs sont en conflit. On obtient un nombre minimal de paires d'amorces lors d'un protocole de « dichotomie » séquentielle, où seule une paire d'amorces est créée à chaque itération, en se fondant sur les résultats des itérations précédentes des expériences. Cette stratégie est irréaliste, car elle mène à un nombre excessif d'itérations. Une alternative, le protocole « à une seule itération », crée toutes les paires d'amorces possibles en un seul tour, aboutissant ainsi à un nombre excessivement grand d'amorces. Comme ces critères sont en conflit, l'ExonPCR cherche un compromis entre la stratégie de la dichotomie et celle de l'itération unique.

9.6 Le jeu des vingt questions avec les gènes

Dans sa forme la plus simple, le problème peut être formulé de la façon suivante : étant donné un ensemble (inconnu) I d'entiers dans l'intervalle $[1, n]$, reconstruire l'ensemble I en posant le minimum de questions de la forme « tel intervalle contient-il un entier de I ? » Dans cette formulation, l'intervalle $[1, n]$ correspond à l'ADNc, I aux bords des exons dans l'ADNc et les questions aux réactions de la PCR définies par une paire d'amorces. Une approche non-adaptative (et triviale) à ce problème consiste à produire n questions sur une seule position : l'intervalle $[i, i]$ contient-il un entier de I ? Dans une approche adaptative, les questions sont produites en itérations tenant compte des résultats de toutes les requêtes précédentes (une seule question est posée à chaque itération).

Dans un souci de simplicité, on considère le cas où l'on connaît le nombre de bords d'exons k. Pour $k = 1$, l'algorithme optimal pour ce problème nécessite au moins $\log n$ questions et est semblable à la recherche binaire (Cormen *et al.*, 1989 [75]). Pour $k > 1$, il est facile de trouver la borne inférieure du nombre de questions utilisées par tout algorithme pour ce problème, qui utilise le modèle de l'arbre de décision. Ce modèle suppose des calculs séquentiels qui posent une question à la fois. Supposons qu'à chaque sommet de l'arbre de décision, on associe tous les ensembles de k points (k-ensembles) compatibles avec toutes les questions du chemin allant à ce sommet. Comme chaque feuille de l'arbre de décision ne contient qu'un k-ensemble, le nombre de feuilles est C_n^k. Comme

chaque arbre de hauteur h possède au plus 2^h feuilles, la borne inférieure pour la hauteur de l'arbre de décision (binaire) est $h \geq \log C_n^k$. Dans le cas relevant de la biologie avec $k \ll n$, le nombre minimal de questions est d'approximativement $k \log n - k \log k$. Si un biologiste tolère une erreur Δ dans les positions des bords d'exons, la borne inférieure pour le nombre de questions est approximativement égale à $k \log \frac{n}{\Delta} - k \log k$. Les tests calculatoires et expérimentaux de l'ExonPCR ont démontré qu'elle est proche de la borne inférieure théorique et qu'environ trente amorces et trois tours sont nécessaires pour trouver des bords d'exons dans une séquence d'ADNc typique.

9.7 Épissage alternatif et cancer

De récentes études fournissent la preuve que le potentiel oncogène du cancer humain peut être modulé par l'épissage alternatif. Par exemple, la progression du cancer de la prostate d'une tumeur androgéno-sensible vers une tumeur androgéno-insensible s'accompagne d'un changement dans l'épissage alternatif du récepteur 2 du facteur de croissance du fibroblaste (Carstens *et al.*, 1997 [59]). Dans une autre étude, Heuze *et al.*, 1999 [160] ont caractérisé une remarquable variante d'épissage alternatif pour l'antigène spécifique de la prostate, le plus important marqueur disponible à l'heure actuelle pour diagnostiquer et surveiller les patients atteints du cancer de la prostate. Les questions qui consistent à savoir quelles variantes importantes d'épissage alternatif de ces gènes ou d'autres gènes sont impliquées dans le cancer demeurent ouvertes. De plus, les variantes alternatives connues des gènes impliqués dans le cancer ont été trouvées par hasard dans une étude au cas par cas.

Étant donné un gène, comment peut-on trouver *toutes* les variantes d'épissage alternatif de ce gène ? Le problème est loin d'être simple, car l'épissage alternatif est très fréquent dans les gènes humains (Mironov *et al.*, 1999 [240]) ; en outre, les méthodes calculatoires pour la prédiction de l'épissage alternatif ne sont pas très fiables.

La première tentative systématique pour élucider les variantes d'épissage des gènes impliqués dans le cancer (ovarien) a été entreprise par Hu *et al.*, 1998 [167]. Ils ont proposé une longue RT-PCR pour amplifier l'ARNm sur toute sa longueur et ont trouvé une nouvelle variante d'épissage pour le gène humain MDR1 de résistance multidrogue et la MVP (pour *major vault protein*). Cette méthode est utile pour détecter quelques variantes importantes à l'aide d'amorces fixées, mais elle présente des difficultés pour détecter les variantes rares (car les variantes importantes ne sont pas supprimées). Elle peut également manquer l'identification d'importantes variantes d'épissage qui ne s'amplifient pas avec la paire d'amorces choisie.

Le défi calculatoire consistant à trouver toutes les variantes d'épissage alternatif (*une encyclopédie de l'épissage alternatif* ou *ASE*) peuvent s'expliquer avec l'exemple suivant. Si un gène avec trois exons possède une variante alternative à laquelle il manque un exon intermédiaire, alors certains produits de

la PCR dans la banque d'ADNc différeront de la longueur de cet exon intermédiaire. Par exemple, une paire d'amorces, dont l'une provient du milieu du premier exon et une autre du milieu du dernier exon, donnera deux produits de la PCR qui diffèrent de la longueur de l'exon intermédiaire. Ceci aboutira à la détection des deux variantes d'épissage alternatif.

Évidemment, il s'agit d'une description naïve et simplifiée qui n'est utilisée que pour illustrer la situation. La complexité de ce problème peut être comprise si l'on considère un gène de dix exons avec un site d'épissage glissant alternatif par exon. Dans ce cas, le nombre de variantes d'épissage potentielles est d'au moins 2^{10} et la façon de trouver les variantes présentes dans la cellule n'est pas claire. Le problème réel est même plus compliqué encore, car certaines de ces variantes d'épissage peuvent être rares et difficiles à détecter par amplification PCR.

La figure 9.11 illustre le problème de la construction d'une ASE pour la séquence « génomique » suivante :

'twas brilliant thrilling morning and the slimy hellish lithe doves

gyrated and gambled nimbly in the waves,

dont les variantes d'épissage alternatif « composent » différents ARNm qui sont semblables au célèbre « ARNm » de Lewis Carroll :

't was brillig, and the slithy toves did gyre and gimble in the wabe.

Le graphe de « l'assemblage d'exons » (voir figure 9.11) comporte un nombre de chemins exponentiel, chaque chemin représentant une variante d'épissage potentielle. Le problème est d'imaginer quels chemins correspondent aux véritables variantes d'épissage. Par exemple, on peut vérifier qu'il existe une variante d'épissage combinant les exons potentiels X et Y, représentés par *T WAS BRILLI* et *G, AND THE SL* avec un *couple d'amorces* XY qui traverse X et Y (par exemple, *BRILLIG, AND T*). En pratique, une sonde XY est construite par concaténation des dix derniers nucléotides de l'exon X et des dix premiers nucléotides de l'exon Y. Apparier XY avec une autre amorce (prise à la fin de l'exon Y, par exemple) confirmera ou infirmera l'hypothèse concernant l'existence d'une variante d'épissage qui combine les exons X et Y. Les couples d'amorces nous permettent d'élaguer les arêtes du graphe d'assemblage des exons qui ne sont pas supportées par une preuve expérimentale. Même après avoir coupé certaines arêtes du graphe, cette approche est confrontée au difficile problème qui consiste à décider quels triplets, quadruplets, etc. d'exons peuvent apparaître parmi les gènes de l'épissage alternatif. La figure 9.11 présente un exemple relativement simple d'un graphe d'assemblage d'exons déjà élagué, avec seulement cinq exons potentiels et cinq chemins possibles : ABCDE, ACDE, ABDE, ABCE et ACE. Les seuls couples d'amorces pour la variante ACE sont AC et CE. Cependant, ces couples d'amorces (appariés à d'autres amorces) ne permettent pas de confirmer ou d'infirmer l'existence

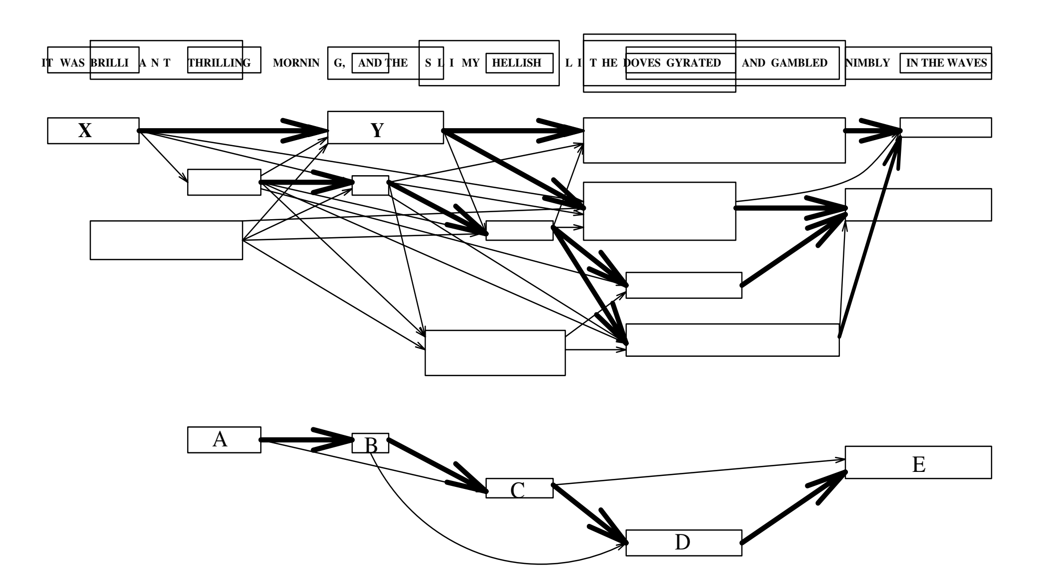


Figure 9.11 – Construction d'une encyclopédie d'épissage alternatif (ASE) à partir d'exons potentiels. Quatre variantes d'épissage différentes (en haut) correspondent à quatre chemins (arêtes dessinées en gras) dans le graphe d'assemblage d'exons. Le nombre total de chemins dans ce graphe est grand et le problème est de savoir comment identifier les chemins qui correspondent à des variantes d'épissage réelles. Le graphe du bas représente le graphe d'assemblage d'exons élagué avec seulement cinq variantes d'épissage potentielles (chemins).

de la variante d'épissage ACE. Ceci est dû au fait que la présence d'un produit de la PCR amplifié par une amorce impliquant AC (par exemple) ne garantit pas la présence de la variante ACE ; en effet, ce produit peut provenir de la variante alternative ACBD. De la même façon, l'amorce CE peut amplifier une variante d'épissage ABCE. Si l'on a de la chance, on peut observer un produit ACE de la PCR relativement court, mais ça ne sera pas le cas si ACE est une variante plutôt rare. La solution serait donnée en formant une paire de couples d'amorces impliquant *à la fois* AC et CE. Cette paire d'amorces amplifie ACE, mais elle n'amplifie aucune autre variante d'épissage dans la figure 9.11.

Les paires d'amorces qui amplifient la variante X mais pas Y sont appelées des paires X+Y. On peut utiliser des paires X+Y pour détecter une variante rare d'épissage X située à l'arrière-plan d'une importante variante d'épissage Y. Cependant, le problème qui consiste à créer un protocole expérimental et calculatoire fiable pour trouver toutes les variantes alternatives n'est pas résolu.

9.8 Quelques autres problèmes et approches

9.8.1 Modèles de Markov cachés pour la prédiction génétique

Le procédé de démolition d'une séquence d'ADN en gènes peut être comparé au procédé d'analyse d'une phrase en différentes parties grammaticales. Searls et Dong, 1993 [312] sont allés plus loin dans cette comparaison plutôt naïve, en préconisant une approche linguistique de la découverte de gènes. Ce concept a été développé ultérieurement dans l'approche des modèles de Markov cachés pour la prédiction génétique (Krogh *et al.*, 1994 [209]) et a mené au programme GENSCAN (Burge et Karlin, 1997 [54]). Les modèles de Markov cachés (notés HMM) pour la découverte de gènes sont constitués de nombreux blocs, chacun d'eux reconnaissant une certaine caractéristique statistique. Par exemple, les profils HMM peuvent être utilisés pour modéliser les sites accepteurs et donneurs. Les statistiques des codons peuvent être captées par un HMM différent qui utilise des codons starts comme état *départ*, des codons comme états intermédiaires et un codon stop comme état fin. Ces HMM peuvent être combinés comme dans l'algorithme GENSCAN de Burge et Karlin, 1997 [54]. Dans une approche voisine, Iseli *et al.*, 1999 [176] ont développé l'algorithme ESTScan pour la prédiction génétique dans EST.

9.8.2 Prédiction génétique bactérienne

Borodovsky *et al.*, 1986 [43] ont été les premiers à appliquer les chaînes de Markov à la prédiction génétique bactérienne. De nombreux projets de séquençage bactérien ont suscité un nouveau défi calculatoire : la prédiction génétique *in silico*, en l'absence de toute analyse expérimentale. Cependant, en l'absence de vérification expérimentale des gènes, il n'y a pas d'échantillons de tests

positifs ou négatifs grâce auxquels on pourrait apprendre les paramètres statistiques pour les régions codantes et non-codantes. Frishman *et al.*, 1998 [113] ont proposé l'approche « similitude d'abord », qui commence par trouver dans l'ADN bactérien des fragments étroitement liés aux fragments d'une base de données, et qui les utilise ensuite comme ensemble d'entraînement initial pour l'algorithme. Après avoir trouvé les paramètres statistiques pour les gènes avec des séquences proches, on les utilise pour la prédiction d'autres gènes de façon itérative. Actuellement, GenMark (Hayes et Borodovsky, 1998 [157]), Glimmer (Salzberg *et al.*, 1998 [295]) et Orpheus (Frishman *et al.*, 1998 [113]) combinent l'approche fondée sur les similitudes et celle qui utilise les statistiques.

Chapitre 10

Réarrangements génomiques

10.1 Introduction

Comparaison génomique et comparaison génétique À la fin des années 80, Jeffrey Palmer et ses collègues ont découvert un modèle remarquable et original de changement évolutif dans les organelles plantigrades. Ils ont comparé les génomes mitochondriaux de *Brassica oleracea* (un chou) et de *Brassica campestris* (un navet), qui sont très étroitement liés (de nombreux gènes sont identiques à 99%). À leur grande surprise, ces molécules, qui sont presqu'identiques dans les *séquences* génétiques, diffèrent considérablement dans l'*ordre* génétique (figure 10.1). Cette découverte, ajoutée à de nombreuses autres études de cette décennie, a prouvé de façon convaincante que les réarrangements génomiques représentaient un mode courant de l'évolution moléculaire.

Chaque étude sur les réarrangements génomiques implique la résolution d'une énigme combinatoire afin de trouver une série de *réarrangements* qui transforme un génome en un autre. La figure 10.1 montre trois réarrangements de ce type, qui « transforment » un chou en un navet. La figure 1.5 présente un *scénario de réarrangement* plus complexe, dans lequel le chromosome X de la souris est transformé en chromosome X humain. L'extrême conservation des gènes sur les chromosomes X à travers les espèces mammifères (Ohno, 1967 [255]) donne l'opportunité d'étudier l'histoire évolutive du chromosome X indépendemment du reste des génomes. Selon la loi d'Ohno, le contenu génétique des chromosomes X a à peine changé au cours du développement mammifère durant les dernières 125 millions d'années. Cependant, l'ordre des gènes sur les chromosomes X a été rompu plusieurs fois.

Il n'est pas facile de vérifier que les six événements évolutifs de la figure 1.5 représentent l'une des *plus courtes* séries d'*inversions* transformant l'ordre génétique de la souris en celui de l'être humain sur le chromosome X. Trouver l'une des plus courtes séries d'inversions entre l'ordre des gènes des ADN mitochondriaux du ver *Ascaris suum* et de l'être humain représente un défi calculatoire encore plus difficile (figure 10.2).

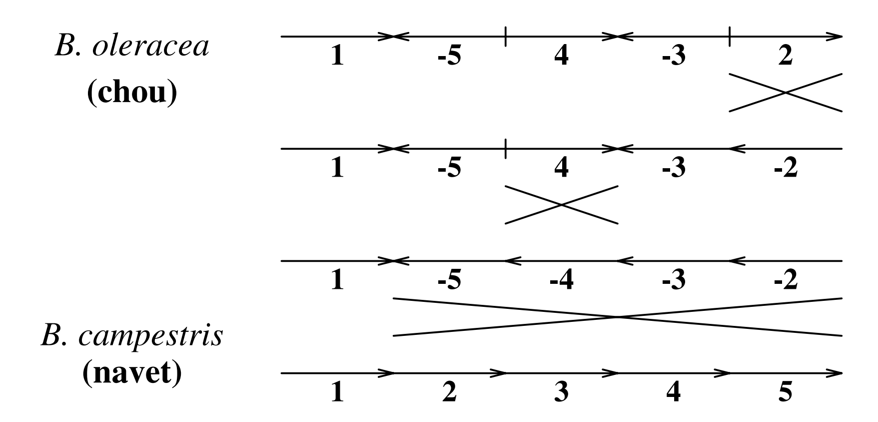


Figure 10.1 – « Transformation » d'un chou en navet.

Dans le cas de génomes constitués d'un petit nombre de « blocs conservés », Palmer et ses collègues ont pu trouver les scénarios de réarrangement les plus économes. Cependant, pour les génomes constitués de plus de dix blocs, la recherche exhaustive de toutes les solutions possibles donne de moins bons résultats que des méthodes « papier et crayon ». Par suite, Palmer et Herbon, 1988 [259] et Makaroff et Palmer, 1988 [229] ont négligé les scénarios de réarrangement les plus économes dans les cas les plus compliqués, tels que navet contre sénevé ou navet contre radis.

La technique d'évolution moléculaire traditionnelle est une comparaison *génétique*, dans laquelle les arbres phylogénétiques sont reconstruits en se basant sur les mutations ponctuelles d'un seul gène (ou d'un petit nombre de gènes). Dans le cas « chou et navet », l'approche qui consiste à comparer des gènes est peu appropriée, car le taux de mutations ponctuelles dans les gènes mitochondriaux du chou et du navet est si faible que leurs gènes sont presque identiques. La *comparaison de génomes* (c'est-à-dire la comparaison des ordres des gènes) est la méthode de choix dans le cas de génomes évoluant très lentement. L'évolution de virus qui se développent rapidement est un autre exemple pour lequel la comparaison de génomes peut être plus concluante que la comparaison de gènes.

Les études sur l'évolution moléculaire des virus de l'herpès ont fait naître beaucoup plus de questions qu'ils n'en ont résolues. Les génomes des virus de l'herpès évoluent si rapidement que les extrêmes des phénotypes actuels peuvent apparaître complètement sans rapport ; la similitude entre de nombreux gènes dans les virus de l'herpès est si faible qu'elle est fréquemment indiscernable du bruit de fond. Par conséquent, les méthodes classiques de comparaison de séquences ne sont pas très utiles pour de tels génomes hautement divergents ; s'aventurer dans la fondrière de la phylogénie moléculaire des virus de l'herpès peut mener à des contradictions, car des gènes différents donnent lieu à des arbres d'évolution différents. Les virus de l'herpès possèdent de 70 à 200 gènes

environ ; ils ont tous en commun sept blocs conservés qui sont réarrangés dans les génomes de différents virus herpès. La figure 10.3 présente différents arrangements de ces blocs dans le cytomégalovirus (CMV) et le virus Epstein-Barr (EBV), ainsi que l'une des plus courtes séries d'inversions transformant l'ordre des gènes CMV en celui des gènes EBV (Hannenhalli *et al.*, 1995 [152]). Le nombre de tels réarrangements à grande échelle (cinq inversions) est nettement inférieur au nombre de mutations ponctuelles entre le CMV et l'EBV (des centaines de milliers). Par conséquent, l'analyse de tels réarrangements au niveau *génomique* peut compléter l'analyse *génétique* utilisée traditionnellement dans l'évolution moléculaire. La comparaison génomique comporte certains avantages et inconvénients vis-à-vis de la comparaison génétique classique : la comparaison génomique ignore les séquences d'ADN actuelles des gènes, tandis que la comparaison génétique ignore l'ordre des gènes. Le but ultime est de combiner les avantages de la comparaison génomique et ceux de la comparaison génétique en un seul algorithme.

12 31 34 28 26 17 29 4 9 36 18 35 19 1 16 14 32 33 22 15 11 27 5 20 13 30 23 10 6 3 24 21 8 25 2 7
20 5 27 11 15 22 33 32 14 16 1 19 35 18 36 9 4 29 17 26 28 34 31 12 13 30 23 10 6 3 24 21 8 25 2 7
1 16 14 32 33 22 15 11 27 5 20 19 35 18 36 9 4 29 17 26 28 34 31 12 13 30 23 10 6 3 24 21 8 25 2 7
1 16 15 22 33 32 14 11 27 5 20 19 35 18 36 9 4 29 17 26 28 34 31 12 13 30 23 10 6 3 24 21 8 25 2 7
1 16 15 36 18 35 19 20 5 27 11 14 32 33 22 9 4 29 17 26 28 34 31 12 13 30 23 10 6 3 24 21 8 25 2 7
1 16 15 14 11 27 5 20 19 35 18 36 32 33 22 9 4 29 17 26 28 34 31 12 13 30 23 10 6 3 24 21 8 25 2 7
1 16 15 14 31 34 28 26 17 29 4 9 22 33 32 36 18 35 19 20 5 27 11 12 13 30 23 10 6 3 24 21 8 25 2 7
1 26 28 34 31 14 15 16 17 29 4 9 22 33 32 36 18 35 19 20 5 27 11 12 13 30 23 10 6 3 24 21 8 25 2 7
1 26 28 18 36 32 33 22 9 4 29 17 16 15 14 31 34 35 19 20 5 27 11 12 13 30 23 10 6 3 24 21 8 25 2 7
1 26 28 29 4 9 22 33 32 36 18 17 16 15 14 31 34 35 19 20 5 27 11 12 13 30 23 10 6 3 24 21 8 25 2 7
1 26 28 29 30 13 12 11 27 5 20 19 35 34 31 14 15 16 17 18 36 32 33 22 9 4 23 10 6 3 24 21 8 25 2 7
1 26 11 12 13 30 29 28 27 5 20 19 35 34 31 14 15 16 17 18 36 32 33 22 9 4 23 10 6 3 24 21 8 25 2 7
1 26 27 28 29 30 13 12 11 5 20 19 35 34 31 14 15 16 17 18 36 32 33 22 9 4 23 10 6 3 24 21 8 25 2 7
1 26 27 28 29 30 31 34 35 19 20 5 11 12 13 14 15 16 17 18 36 32 33 22 9 4 23 10 6 3 24 21 8 25 2 7
1 26 27 28 29 30 31 34 35 19 20 9 22 33 32 36 18 17 16 15 14 13 12 11 5 4 23 10 6 3 24 21 8 25 2 7
1 26 27 28 29 30 31 22 9 20 19 35 34 33 32 36 18 17 16 15 14 13 12 11 5 4 23 10 6 3 24 21 8 25 2 7
1 26 27 28 29 30 31 32 33 34 35 19 20 9 22 36 18 17 16 15 14 13 12 11 5 4 23 10 6 3 24 21 8 25 2 7
1 26 27 28 29 30 31 32 33 34 35 36 22 9 20 19 18 17 16 15 14 13 12 11 5 4 23 10 6 3 24 21 8 25 2 7
1 26 27 28 29 30 31 32 33 34 35 36 22 9 24 3 6 10 23 4 5 11 12 13 14 15 16 17 18 19 20 21 8 25 2 7
1 26 27 28 29 30 31 32 33 34 35 36 22 9 8 21 20 19 18 17 16 15 14 13 12 11 5 4 23 10 6 3 24 25 2 7
1 26 27 28 29 30 31 32 33 34 35 36 8 9 22 21 20 19 18 17 16 15 14 13 12 11 5 4 23 10 6 3 24 25 2 7
1 26 27 28 29 30 31 32 33 34 35 36 8 9 22 21 20 19 18 17 16 15 14 13 12 11 5 4 3 6 10 23 24 25 2 7
1 26 27 28 29 30 31 32 33 34 35 36 8 9 22 21 20 19 18 17 16 15 14 13 12 11 5 4 3 2 25 24 23 10 6 7
1 2 3 4 5 11 12 13 14 15 16 17 18 19 20 21 22 9 8 36 35 34 33 32 31 30 29 28 27 26 25 24 23 10 6 7
1 2 3 4 5 11 12 13 14 15 16 17 18 19 20 21 22 9 8 7 6 10 23 24 25 26 27 28 29 30 31 32 33 34 35 36
1 2 3 4 5 6 7 8 9 22 21 20 19 18 17 16 15 14 13 12 11 10 23 24 25 26 27 28 29 30 31 32 33 34 35 36
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

Figure 10.2 – Un des scénarios de réarrangement les plus économiques pour la transformation de l'ADN mitochondrial du ver *Ascaris Suum* en l'ADN mitochondrial humain (26 inversions).

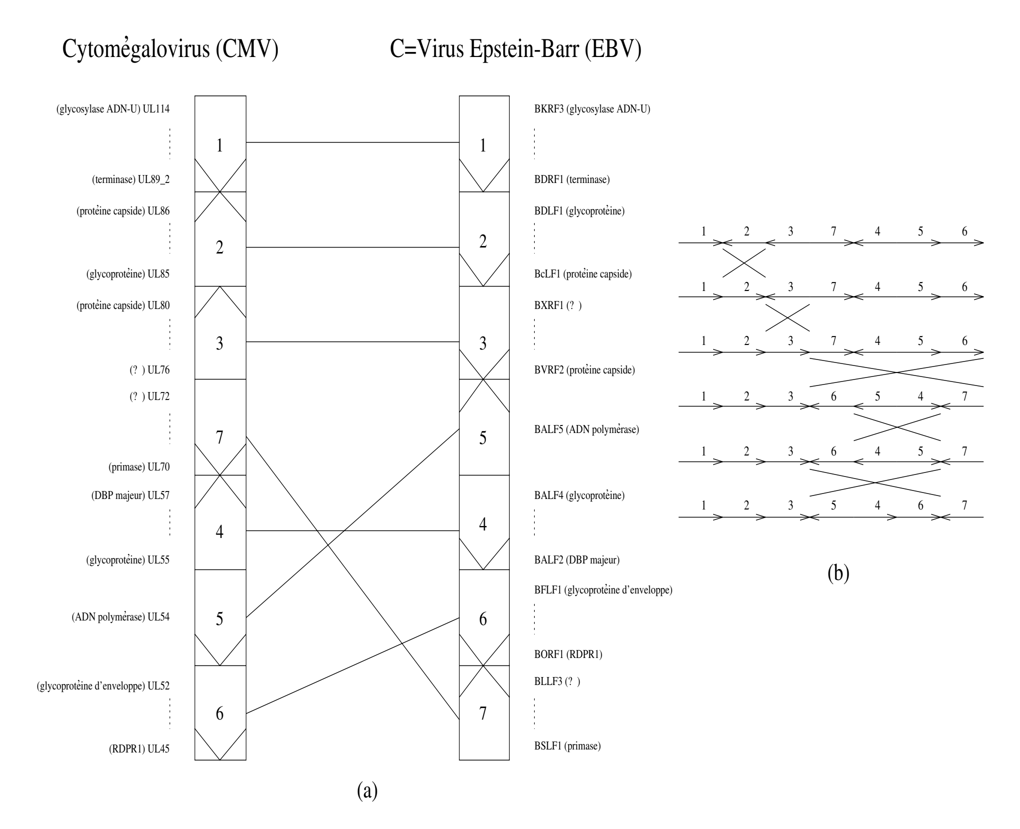


Figure 10.3 – Organisation comparée des génomes du CMV et de l'EBV (a) et plus courte série de réarrangements transformant l'ordre génétique du CMV en celui de l'EBV (b).

L'analyse des réarrangements génomiques en biologie moléculaire a débuté à la fin des années 30 avec Dobzhansky et Sturtevant, qui ont publié un article de première importance présentant un scénario de réarrangement avec 17 inversions pour les espèces de mouches à fruits *Drosophila* (Dobzhansky et Sturtevant, 1938 [87]). Avec l'avènement de la cartographie et du séquençage à grande échelle, le nombre de problèmes de *comparaison de génomes* a augmenté rapidement dans différents domaines, notamment au niveau de l'évolution des virus, des bactéries, des levures, des plantes et des animaux.

Tri par inversions Une approche calculatoire fondée sur la comparaison de l'ordre des gènes a été introduite par David Sankoff (Sankoff *et al.*, 1990, 1992 [302, 304] et Sankoff, 1992 [300]). Les réarrangements génomiques peuvent être vus comme une sorte de tri par inversions, comme décrit ci-dessous. L'ordre des gènes dans deux organismes est représenté par des permutations $\pi = \pi_1\pi_2\ldots\pi_n$ et $\sigma = \sigma_1\sigma_2\ldots\sigma_n$. L'*inversion* $\rho(i,j)$ de l'intervalle $[i,j]$ corres-

pond à la permutation

$$\begin{pmatrix} 1 & 2 & \ldots & i-1 & \mathbf{i} & \mathbf{i+1} & \ldots & \mathbf{j-1} & \mathbf{j} & j+1 & \ldots & n \\ 1 & 2 & \ldots & i-1 & \mathbf{j} & \mathbf{j-1} & \ldots & \mathbf{i+1} & \mathbf{i} & j+1 & \ldots & n \end{pmatrix}$$

Il est clair que $\rho(i,j)$ a pour effet d'inverser l'ordre de $\pi_i \pi_{i+1} \ldots \pi_j$ et de transformer $\pi_1 \ldots \pi_{i-1} \pi_i \ldots \pi_j \pi_{j+1} \ldots \pi_n$ en $\pi \cdot \rho(i,j) = \pi_1 \ldots \pi_{i-1} \pi_j \ldots \pi_i$ $\pi_{j+1} \ldots \pi_n$.

Étant données des permutations π et σ, le *problème de la distance d'inversion* est de trouver une suite d'inversions $\rho_1, \rho_2, \ldots, \rho_t$ qui vérifie $\pi \cdot \rho_1 \cdot \rho_2 \cdots \rho_t = \sigma$ avec t minimal. L'entier t est appelé la *distance d'inversion* entre π et σ. Le *tri de π par inversions* est le problème qui revient à trouver la distance d'inversion $d(\pi)$ entre π et la permutation identité $(1\,2 \ldots n)$.

Les informaticiens ont étudié un problème voisin, le *tri par inversions de préfixes* (également connu sous le nom de *tri du crêpier*) : étant donnée une permutation π arbitraire, trouver $d_{pref}(\pi)$, le nombre minimal d'inversions de la forme $\rho(1,i)$ nécessaires pour trier π. Le problème de l'arrangement des crêpes provient d'une situation de la « vie quotidienne » décrite par Harry Dweigter :

Le cuisinier dont nous parlons est négligent et, quand il prépare un tas de crêpes, elles sont toutes de tailles différentes. Par conséquent, lorsque je les sers à un client, je les réarrange (de sorte que les plus petites se trouvent au-dessus, et ainsi de suite, jusqu'à avoir les plus grandes en dessous) en en prenant quelques unes du dessus et en les passant en dessous ; je répète ceci (en variant le nombre de crêpes déplacées) autant de fois que nécessaire. S'il y a n crêpes, quel nombre maximal de mouvements aurai-je à réaliser pour les réarranger ?

Bill Gates (étudiant à Harvard à la fin des années 70, maintenant chez Microsoft) et Cristos Papadimitriou ont été les premiers à tenter de résoudre ce problème (Gates et Papadimitriou, 1979 [120]). Ils ont prouvé que le *diamètre d'inversion de préfixes* du groupe symétrique, $d_{préf}(n) = \max_{\pi \in S_n} d_{préf}(\pi)$, est inférieur ou égal à $\frac{5}{3}n + \frac{5}{3}$ et que, pour une infinité de valeurs de n, on a $d_{préf}(n) \geq \frac{17}{16}n$. Le problème de l'arrangement des crêpes n'a toujours pas été résolu.

Le graphe des points de rupture Qu'est-ce qui rend le tri d'une permutation aussi difficile ? Dans les premières études calculatoires des réarrangements de génomes, Watterson *et al.*, 1982 [366] et Nadeau et Taylor, 1984 [248] ont introduit la notion de *point de rupture* et ont remarqué des corrélations entre la distance d'inversion et le nombre de points de rupture (en fait, Sturtevant et Dobzhansky, 1936 [331] ont discuté implicitement de ces corrélations il y a soixante ans !). Ci-dessous, nous définissons la notion de point de rupture.

On écrit $i \sim j$ si $|i-j| = 1$. On étend une permutation $\pi = \pi_1 \pi_2 \ldots \pi_n$ en ajoutant $\pi_0 = 0$ et $\pi_{n+1} = n+1$. Un couple d'éléments $(\pi_i,\ \pi_{i+1})$ de π, avec $0 \leq i \leq n$, est appelée une *adjacence* s'il vérifie $\pi_i \sim \pi_{i+1}$; sinon, on dit que c'est un *point de rupture* (figure 10.4). Comme la permutation identité ne possède pas de point de rupture, le tri par inversions consiste en l'élimination

des points de rupture. En remarquant que toute inversion peut éliminer *au plus* deux points de rupture, on en déduit immédiatement l'inégalité $d(\pi) \geq \frac{b(\pi)}{2}$, où $b(\pi)$ est le nombre de points de rupture dans π. En se fondant sur la notion de point de rupture, Kececioglu et Sankoff, 1995 [194] ont trouvé un algorithme d'approximation pour le tri par inversions avec une garantie de performance égale à 2. Ils ont également imaginé des bornes efficaces en résolvant le problème de la distance d'inversion de façon presque optimale pour n allant de 30 à 50. Cet intervalle couvre le cas biologiquement important des génomes mitochondriaux animaux.

Cependant, l'estimation de la distance d'inversion en termes de points de rupture est très imprécise. Bafna et Pevzner, 1996 [19] ont montré qu'un autre paramètre (la taille d'une décomposition cyclique maximale du graphe des points de rupture) estime la distance d'inversion avec une plus grande précision.

Le *graphe des points de rupture* d'une permutation π est un graphe $G(\pi)$ aux arêtes coloriées qui possède $n+2$ sommets $\{\pi_0, \pi_1, \ldots, \pi_n, \pi_{n+1}\} \equiv \{0, 1, \ldots, n, n+1\}$. On relie les sommets π_i et π_{i+1} par une arête *noire* pour $0 \leq i \leq n$. On relie les sommets π_i et π_j par une arête *grise* s'ils vérifient $\pi_i \sim \pi_j$. La figure 10.4 suggère qu'un graphe des points de rupture s'obtient en superposant un chemin noir qui traverse les sommets $0, 1, \ldots, n, n+1$ dans l'ordre donné par la permutation π et un chemin gris qui passe par les sommets dans l'ordre donné par la permutation identité.

Dans un graphe G, un *cycle* dont les arêtes sont coloriées est dit *alterné* si les couleurs de deux arêtes consécutives de ce cycle sont distinctes. Par la suite, lorsque nous évoquerons un cycle, ceci signifiera implicitement qu'il est alterné. Dans un graphe G, un sommet v est dit *équilibré* si le nombre d'arêtes noires incidentes à v est égal au nombre d'arêtes grises incidentes à v ; un *graphe équilibré* est un graphe dont chaque sommet est équilibré. Il est clair que $G(\pi)$ est un graphe équilibré : par conséquent, il contient un cycle eulérien alterné. Il existe donc une *décomposition cyclique* de $G(\pi)$ en cycles alternés aux arêtes disjointes (chaque arête du graphe appartient à exactement un cycle de la décomposition). Un cycle d'une décomposition en arêtes peut s'auto-intersecter (cycle non élémentaire). Le graphe des points de rupture dans la figure 10.4 peut être décomposé en quatre cycles, dont l'un s'auto-intersecte. On s'intéresse à la décomposition du graphe des points de rupture en un nombre *maximal* $c(\pi)$ de cycles alternés aux arêtes disjointes. Pour la permutation de la figure 10.4, on a $c(\pi) = 4$.

Les décompositions cycliques jouent un rôle important dans l'estimation de la distance d'inversion. Lorsque l'on applique une inversion à une permutation, le nombre de cycles dans une décomposition maximale peut changer d'au plus un (alors que le nombre de points de rupture peut changer de deux). Bafna et Pevzner, 1996 [19] ont prouvé la borne $d(\pi) \geq n + 1 - c(\pi)$, qui est plus précise que celle en termes de points de rupture, $d(\pi) \geq b(\pi)/2$. Pour la plupart des exemples à caractère biologique, on a $d(\pi) = n+1-c(\pi)$, réduisant ainsi le problème de la distance d'inversion à celui de la décomposition cyclique maximale.

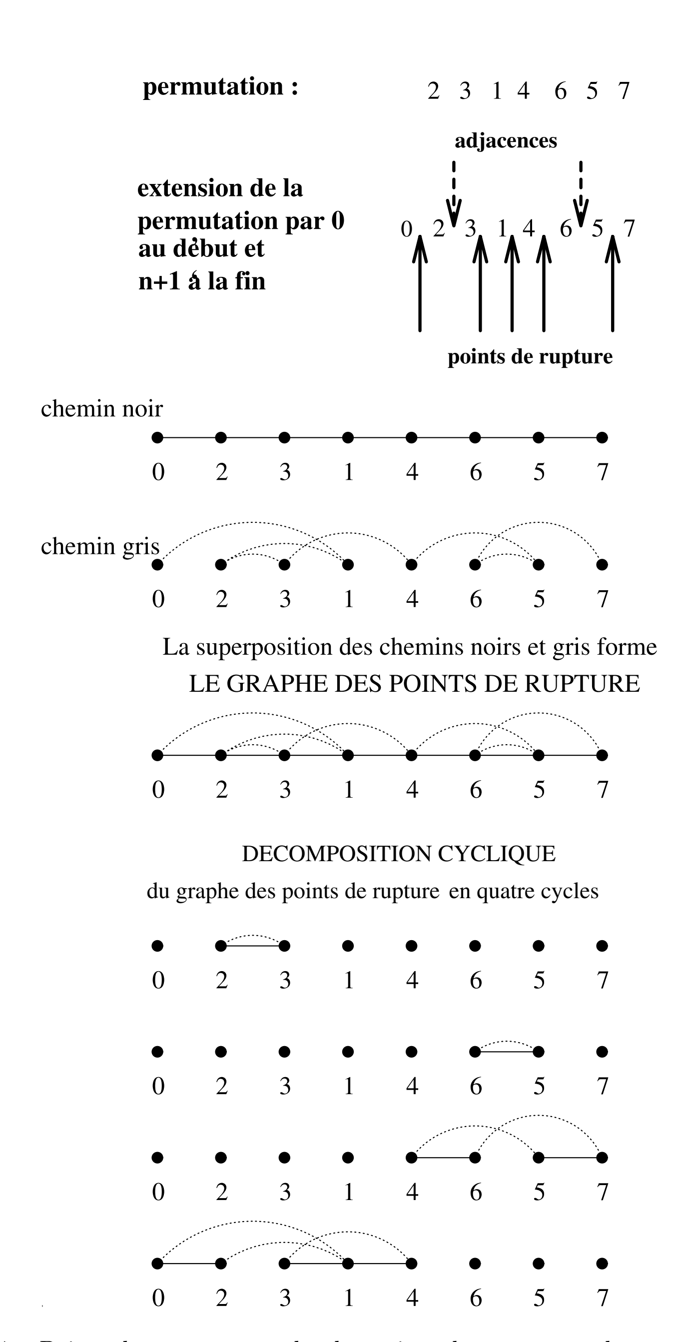


Figure 10.4 – Points de rupture, graphe des points de rupture et décomposition cyclique maximale.

Théorème de dualité pour les permutations signées Trouver une décomposition cyclique maximale est un problème difficile. Heureusement, dans le cas des *permutations signées*, plus applicable à la biologie, ce problème est trivial. Les gènes sont des fragments d'ADN *orientés* et une séquence de n gènes dans un génome est représentée par une permutation *signée* sur $\{1, \ldots n\}$, avec un signe + ou − associé à chaque élément de π. Par exemple, l'ordre des gènes de *B. oleracea* présenté dans la figure 10.1 est modélisé par la permutation signée $(+1, -5, +4, -3, +2)$. Dans le cas signé, toute inversion de fragment $[i, j]$ change l'ordre et les signes des éléments à l'intérieur de ce fragment (figure 10.1). On s'intéresse au nombre minimal $d(\pi)$ d'inversions nécessaires pour transformer une permutation signée π en la permutation signée identité $(+1, +2, \ldots, +n)$.

Bafna et Pevzner, 1996 [19] ont remarqué que le concept de graphe des points de rupture s'étend naturellement aux permutations signées en dédoublant chaque élément orienté i par deux éléments non orientés i_a et i_b, qui se substituent aux extrémités intiale et finale de l'élément orienté i (figure 10.5).

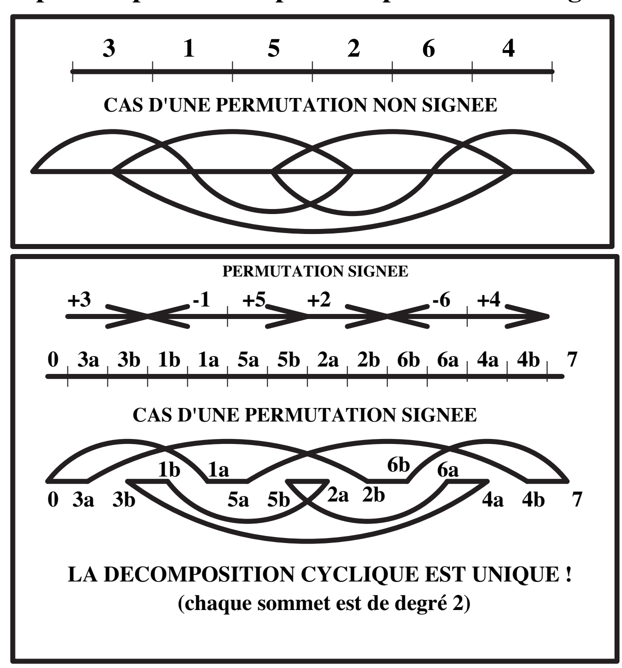


Figure 10.5 – Modélisation d'une permutation signée par une permutation non signée.

Pour les permutations signées, la borne $d(\pi) \geq n+1-c(\pi)$ donne une très bonne valeur approchée de la distance d'inversion, que les données soient simulées ou biologiques. On peut alors se demander si la borne $d(\pi) \geq n+1-c(\pi)$ n'omet pas un autre paramètre (en plus de la taille d'une décomposition cyclique maximale) qui permettrait de combler l'écart entre $d(\pi)$ et $n+1-c(\pi)$. Hannenhalli et Pevzner, 1995 [154] ont découvert un autre paramètre « caché » (nombre d'*obstacles* dans π) qui complique le tri d'une permutation signée ; ils ont montré :

$$n+1-c(\pi)+h(\pi) \leq d(\pi) \leq n+2-c(\pi)+h(\pi), \tag{10.1}$$

où $h(\pi)$ est le nombre d'obstacles dans π. Ils ont également prouvé le théorème de dualité pour les permutations signées et développé un algorithme polynomial pour calculer $d(\pi)$.

Permutations non signées et cartographie physique comparative Le tri par inversions de permutations (non signées) étant NP-complet (Caprara, 1997 [57]), de nombreux chercheurs ont essayé d'imaginer un algorithme d'approximation pratique pour trier (des permutations non signées) par inversions.

Un *bloc* de π est un intervalle $\pi_i \dots \pi_j$ qui ne contient aucun point de rupture, autrement dit (π_k, π_{k+1}) est une adjacence pour $0 \leq i \leq k < j \leq n+1$. On définit une *bande* de π comme étant un bloc maximal, autrement dit un bloc $\pi_i \dots \pi_j$ tel que (π_{i-1}, π_i) et (π_j, π_{j+1}) sont des points de rupture. Une bande composée d'un seul élément est appelée un *singleton*, une bande de deux éléments est une *2-bande* et une bande de plus de deux éléments est dite *longue*. Il s'avère que les singletons sont à l'origine d'un défi majeur dans le tri par inversions de permutations non signées.

Une inversion $\rho(i,j)$ *coupe* une bande $\pi_k \dots \pi_l$ si elle vérifie $k < i \leq l$ ou $k \leq j < l$. Une inversion qui coupe une bande sépare des éléments qui sont consécutifs dans la permutation identité. Par conséquent, il est naturel d'espérer que, pour toute permutation π, il existe un tri par inversions (optimal) de π qui ne coupe pas de bande. Cependant, ceci est faux. La permutation 3412 nécessite trois inversions si l'on ne veut pas couper de bande ; pourtant, elle peut être triée grâce à deux inversions : $3412 \rightarrow 1432 \rightarrow 1234$. Kececioglu et Sankoff, 1993 [192] ont conjecturé que chaque permutation possède un tri par inversions optimal qui ne coupe pas de longue bande et qui n'augmente pas le nombre de points de rupture. Comme la permutation identité ne possède aucun point de rupture, le tri par inversions correspond à l'élimination des points de rupture. Il est ainsi naturel d'espérer que, pour toute permutation, il existe un tri par inversions optimal qui n'augmente jamais le nombre de points de rupture. Grâce au théorème de dualité pour les permutations signées, Hannenhalli et Pevzner, 1996 [155] ont prouvé que : « les inversions ne coupent pas les longues bandes » et « les inversions n'augmentent pas le nombre de points de rupture ».

Les biologistes trouvent les ordres des gènes soit en séquençant des génomes entiers, soit en utilisant la cartographie physique comparative. Le séquençage fournit de l'information sur les directions des gènes et permet de représenter un

Cartes physiques comparatives du chou et du navet

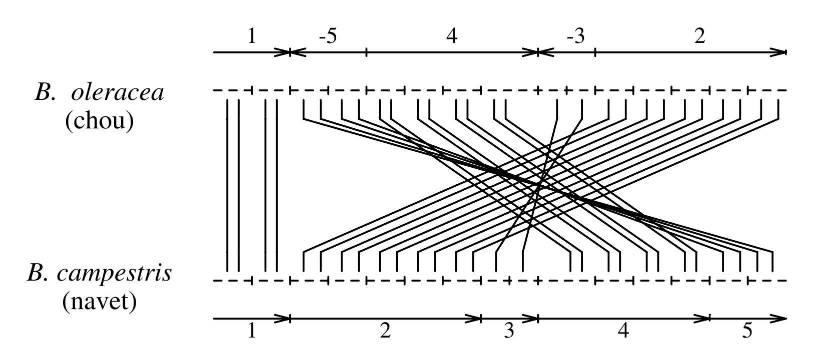


Figure 10.6 – La carte physique comparative du chou et du navet (permutation non signée) et la permutation signée correspondante.

génome par une permutation signée. Cependant, le séquençage de génomes entiers est encore coûteux et la plupart des données expérimentales actuellement disponibles sur les ordres des gènes sont fondées sur les cartes physiques comparatives. Les cartes physiques n'apportent généralement pas d'information sur les directions des gènes et, par conséquent, elles aboutissent à la représentation d'un génome sous la forme d'une permutation *non signée* π. Les biologistes tentent de trouver une permutation signée à partir de cette représentation, en associant un signe positif (resp. négatif) aux bandes croissantes (resp. décroissantes) de π (figure 10.6). La propriété selon laquelle « les inversions ne coupent pas les longues bandes » fournit une justification théorique pour une telle procédure dans le cas de longues bandes. Dans le même temps, pour les 2-bandes, cette procédure peut ne pas parvenir à trouver un scénario de réarrangement optimal. Hannenhalli et Pevzner, 1996 [155] ont donné un exemple biologique pour lequel cette procédure ne fonctionne pas et ont décrit un algorithme réglant ce problème.

La difficulté de l'analyse des permutations sans singleton se résume en l'alternative suivante : *« couper ou ne pas couper »* les 2-bandes ? Une caractérisation d'un ensemble de 2-bandes *« à couper »* (Hannenhalli et Pevzner, 1996 [155]) mène à un algorithme polynomial pour le tri de permutations sans singleton et à un algorithme polynomial pour le tri de permutations ayant un petit nombre de singletons. L'algorithme peut s'appliquer à l'analyse des scénarios de réarrangement imaginés à partir des cartes physiques comparatives.

Les cartes physiques de faible résolution contiennent habituellement de nombreux singletons et, par suite, les scénarios de réarrangement pour de telles cartes sont difficiles à analyser. L'algorithme de Hannenhalli et Pevzner, 1996 [155] a une complexité temporelle polynomiale si le nombre de singletons est en $O(\log n)$. Ceci suggère que $O(\log n)$ singletons est l'option de résolution désirée pour la cartographie physique comparative dans les études sur l'évo-

lution moléculaire. Si le nombre de singletons est grand, un biologiste pourra choisir des expériences supplémentaires (c'est-à-dire le séquençage de certains secteurs) pour résoudre les ambiguïtés sur les directions des gènes.

Réarrangements de génomes multichromosomiques Quand les frères Grimm ont décrit la transformation d'un homme en souris dans le conte intitulé « le Chat botté », ils auraient difficilement pu anticiper que, deux siècles plus tard, les humains et les souris deviendraient les deux mammifères les plus étudiés en génétique. La cartographie physique comparative homme-souris a débuté il y a vingt ans et, actuellement, quelques milliers de paires de gènes homologues sont cartographiés chez ces espèces. Par la suite, les biologistes ont trouvé que les gènes voisins chez l'homme et la souris ne sont pas distribués de façon chaotique parmi les génomes, mais qu'ils forment plutôt des « blocs conservés ». Les données actuelles en cartographie comparative indiquent que le génome humain et celui de la souris contiennent approximativement 150 blocs qui sont « brassés » de façon comparable chez les humains et les souris (Copeland *et al.*, 1993 [74]). Par exemple, le chromosome 7 chez l'humain peut être vu comme une mosaïque de différents gènes provenant des chromosomes 2, 5, 6, 11, 12 et 13 chez la souris (figure 1.4). Le brassage des blocs se produit très rarement (environ une fois en un million d'années), ce qui donne donc aux biologistes l'espoir de reconstruire un scénario de réarrangement de l'évolution homme-souris. Dans leur article novateur, Nadeau et Taylor, 1984 [248] ont estimé que, de façon surprenante, peu de réarrangements génomiques (178 ± 39) se sont produits depuis la divergence de l'être humain et de la souris il y a 80 millions d'années.

Dans le modèle que l'on considère, chaque gène est représenté par un entier dont le *signe* (« $+$ » ou « $-$ ») reflète la *direction*. Par définition, un *chromosome* est une *suite* de gènes, tandis qu'un *génome* est un *ensemble* de chromosomes. Étant donnés deux génomes Π et Γ, on s'intéresse à l'un des scénarios les plus économes de l'*évolution* de Π en Γ, c'est-à-dire à la plus courte séquence d'événements de réarrangement (voir infra) nécessaires pour transformer Π en Γ. Par la suite, on suppose que Π et Γ contiennent le même ensemble de gènes. La figure 10.7 illustre quatre événements de réarrangement qui transforment un génome en un autre.

Soit $\Pi = \{\pi(1), \ldots, \pi(N)\}$ un génome constitué de N chromosomes et soit $\pi(i) = (\pi(i)_1 \ldots \pi(i)_{n_i})$, n_i étant le nombre de gènes dans le i-ième chromosome. Chaque chromosome π peut être lu soit « de gauche à droite » (c'est-à-dire comme $\pi = (\pi_1 \ldots \pi_n)$), soit de « droite à gauche » (c'est-à-dire comme $-\pi = (-\pi_n \ldots - \pi_1)$), ce qui aboutit à deux représentations équivalentes du même chromosome (i.e. les *directions* des chromosomes ne sont pas pertinentes). Les quatre événements élémentaires de réarrangement les plus fréquents dans les génomes multichromosomiques sont les *inversions*, les *translocations*, les *fusions* et les *fissions*, que nous allons maintenant définir.

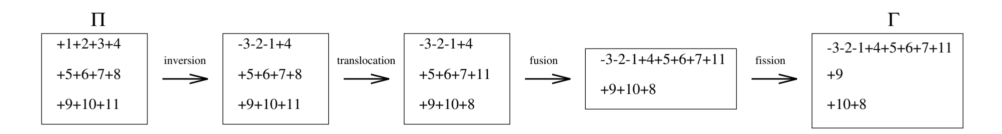


Figure 10.7 – Évolution du génome Π en le génome Γ.

Soient $\pi = \pi_1 \dots \pi_n$ un chromosome et i et j des entiers avec $1 \leq i \leq j \leq n$. Une *inversion* $\rho(\pi, i, j)$ sur un chromosome π réarrange les gènes *à l'intérieur* de $\pi = \pi_1 \dots \pi_{i-1}\pi_i \dots \pi_j\pi_{j+1} \dots \pi_n$ et transforme π en $\pi_1 \dots \pi_{i-1} - \pi_j \dots - \pi_i$ $\pi_{j+1} \dots \pi_n$. Soient $\pi = \pi_1 \dots \pi_n$ et $\sigma = \sigma_1 \dots \sigma_m$ deux chromosomes et i et j des entiers qui vérifient $1 \leq i \leq n+1$, $1 \leq j \leq m+1$. Une *translocation* $\rho(\pi, \sigma, i, j)$ échange les gènes *entre* les chromosomes π et σ et les transforme en les chromosomes $\pi_1 \dots \pi_{i-1}\sigma_j \dots \sigma_m$ et $\sigma_1 \dots \sigma_{j-1}\pi_i \dots \pi_n$, avec $(i-1)+(m-j+1)$ et $(j-1)+(n-i+1)$ gènes respectivement. On désigne par $\Pi \cdot \rho$ le génome obtenu à partir de Π à la suite d'un réarrangement ρ (inversion ou translocation). Étant donnés des génomes Π et Γ, le *problème du tri génomique* consiste à trouver une série d'inversions et de translocations $\rho_1, \dots, \rho_t$ qui vérifient $\Pi \cdot \rho_1 \cdots \rho_t = \Gamma$ avec t minimal. On appelle t la *distance génomique* entre Π et Γ. Le *problème de la distance génomique* est le problème qui consiste à trouver la valeur de $d(\Pi, \Gamma)$ entre Π et Γ.

La translocation $\rho(\pi, \sigma, n+1, 1)$ concatène les chromosomes π et σ, donnant un chromosome $\pi_1 \dots \pi_n\sigma_1 \dots \sigma_m$ et un chromosome *vide* $\emptyset$. Cette translocation spéciale, qui aboutit à une réduction du nombre de chromosomes (non vides), est connue en biologie moléculaire sous le nom de *fusion*. La translocation $\rho(\pi, \emptyset, i, 1)$, pour $1 < i < n$, casse un chromosome π en deux chromosomes $(\pi_1 \dots \pi_{i-1})$ et $(\pi_i \dots \pi_n)$. Cette translocation, qui amène une augmentation du nombre de chromosomes (non vides), est appelée une *fission*. Les fusions et les fissions sont plutôt fréquentes dans l'évolution des mammifères ; par exemple, la différence majeure dans l'organisation globale des génomes des humains et des chimpanzés correspond à la fusion de deux chromosomes de chimpanzés en un chromosome humain.

Kececioglu et Ravi, 1995 [191] ont été les premiers à tenter d'analyser les réarrangements des génomes multichromosomiques. Leur algorithme d'approximation aborde le cas de deux génomes avec le même nombre de chromosomes. Ceci est une sérieuse limite, car différents organismes (en particulier l'homme et la souris) ne possèdent pas le même nombre de chromosomes. De ce point de vue, tout modèle réaliste de réarrangements génomiques devrait inclure des fusions et des fissions. Or il s'avère que celles-ci présentent une difficulté majeure dans l'analyse des réarrangements de génomes. Hannenhalli et Pevzner, 1995 [153] ont prouvé le théorème de dualité pour les génomes multichromosomiques ; il exprime la distance génomique en termes de sept paramètres qui reflètent différentes propriétés combinatoires des ensembles de chaînes. À partir de ce résultat, ils ont trouvé un algorithme polynomial pour ce problème.

L'idée de l'analyse est de concaténer N chromosomes de Π et Γ en des permutations π et γ, respectivement, et d'imiter le tri génomique de Π en Γ en triant par inversions π en γ. La difficulté de cette approche est qu'il existe $N! \times 2^N$ concaténés différents pour Π et Γ et que seuls certains d'entre eux, appelés *concaténés optimaux*, imitent un tri *optimal* de Π en Γ. Hannenhalli et Pevzner, 1995 [153] ont introduit des techniques appelées *retournement* et *coiffage* de chromosomes qui permettent de trouver un concaténé optimal.

Évidemment, les ordres des gènes pour seulement deux génomes suffisent difficilement à définir un scénario de réarrangement correct. La cartographie génétique comparative a rendu possible la création de cartes comparatives pour de nombreuses espèces de mammifères (O'Brien et Graves, 1991 [254]). Cependant, la résolution de ces cartes est sensiblement plus faible que celle de la carte homme-souris. Comme la cartographie physique comparative est plutôt laborieuse, on peut difficilement s'attendre à ce que les énormes efforts fournis pour obtenir la carte humain-souris soient répétés pour les génomes d'autres mammifères. Cependant, une technique expérimentale appelée *peinture chromosomique* permet de trouver l'ordre des gènes sans réellement construire une carte précise « fondée sur les gènes ». Par le passé, les applications de la peinture chromosomique se limitaient aux primates (Jauch *et al.*, 1992 [178]) ; les tentatives pour étendre cette approche à d'autres mammifères étaient inefficaces à cause de la diversité des séquences ADN entre les espèces de parenté éloignée. Par la suite, Scherthan *et al.*, 1994 [307] ont développé une version améliorée de la peinture chromosomique, appelée *ZOO-FISH*, qui est capable de détecter des fragments de chromosomes homologues chez des espèces mammifères distantes. Grâce à ZOO-FISH, Rettenberger *et al.*, 1995 [284] ont rapidement complété le projet de peinture chromosomique humain-cochon et ont identifié 47 blocs conservés communs à l'humain et au cochon. Le succès du projet de peinture chromosomique humain-cochon indique que les ordres des gènes de nombreuses espèces mammifères peuvent être produits par ZOO-FISH à peu de frais, fournissant ainsi une nouvelle source de données inestimable pour s'attaquer au problème de l'évolution des mammifères.

10.2 Le graphe des points de rupture

La décomposition cyclique est une notion plutôt exotique qui, à première vue, a peu de choses en commun avec les réarrangements génomiques. Cependant, l'observation selon laquelle une inversion change d'au plus un le nombre de cycles dans une décomposition maximale permet de limiter les distance d'inversion en termes de décomposition cyclique maximale.

Théorème 10.1 *Pour toute permutation π et toute inversion ρ, on a :* $c(\pi \cdot \rho) - c(\pi) \leq 1$.

Preuve Une inversion arbitraire $\rho(i,j)$ implique quatre sommets du graphe $G(\pi)$ et aboutit au remplacement des deux arêtes noires $SUP = \{(\pi_{i-1}, \pi_i), (\pi_j, \pi_{j+1})\}$ par les arêtes noires $AJOUT = \{(\pi_{i-1}, \pi_j), (\pi_i, \pi_{j+1})\}$.

Si les deux arêtes noires d'$AJOUT$ appartiennent au même cycle dans une décomposition cyclique maximale de $G(\pi \cdot \rho)$, alors la suppression de ce cycle fournit une décomposition cyclique de $G(\pi)$ avec au moins $c(\pi \cdot \rho) - 1$ cycles. Par conséquent, on obtient : $c(\pi) \geq c(\pi \cdot \rho) - 1$.

Par ailleurs, si les arêtes noires de $AJOUT$ appartiennent à deux cycles différents C_1 et C_2 dans une décomposition cyclique maximale de $G(\pi \cdot \rho)$, alors la suppression de $C_1 \cup C_2$ donne un ensemble de cycles à arêtes disjointes de taille $c(\pi \cdot \rho) - 2$ dans le graphe $G(\pi \cdot \rho) \setminus (C_1 \cup C_2)$. Il est clair que l'ensemble d'arêtes $(C_1 \cup C_2 \cup SUP) \setminus AJOUT$ forme un graphe équilibré et doit contenir au moins un cycle. En combinant ce cycle avec les $c(\pi \cdot \rho) - 2$ cycles obtenus précédemment, on obtient une décomposition cyclique de $G(\pi) = (G(\pi \cdot \rho) \setminus (C_1 \cup C_2)) \cup (C_1 \cup C_2 \cup SUP \setminus AJOUT)$ en au moins $c(\pi \cdot \rho) - 1$ cycles. ■

Le théorème 10.1, utilisé avec l'observation selon laquelle $c(\iota)$ vaut $n+1$ pour la permutation identité ι, implique immédiatement $d(\pi) \geq c(\iota) - c(\pi) \equiv n + 1 - c(\pi)$:

Théorème 10.2 *Pour toute permutation π, on a : $d(\pi) \geq n + 1 - c(\pi)$.*

10.3 Permutations « difficiles à trier »

On définit $d(n) = \max_{\pi \in S_n} d(\pi)$ comme étant le *diamètre d'inversion* du groupe symétrique d'ordre n. Gollan a conjecturé qu'il vérifie $d(n) = n - 1$ et que seules une permutation γ_n et sa permutation inverse γ_n^{-1} nécessitent $n - 1$ inversions pour être triées. La permutation de *Gollan*, dans la notation en une ligne, est définie comme suit :

$$\gamma_n = \begin{cases} (3, 1, 5, 2, 7, 4, \ldots, n-3, n-5, n-1, n-4, n, n-2), & \text{pour } n \text{ pair} \\ (3, 1, 5, 2, 7, 4, \ldots, n-6, n-2, n-5, n, n-3, n-1), & \text{pour } n \text{ impair.} \end{cases}$$

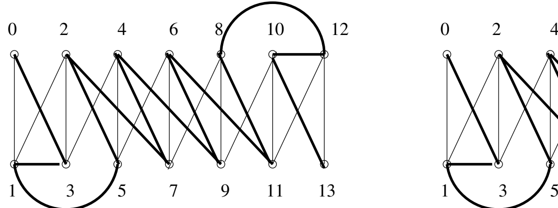
Bafna et Pevzner, 1996 [19] ont prouvé la conjecture de Gollan en montrant l'égalité $c(\gamma_n) = 2$ et en appliquant le théorème 10.2. Par la suite, ils ont démontré que la distance d'inversion entre deux permutations aléatoires était très proche du diamètre d'inversion du groupe symétrique ; la distance d'inversion fournit donc une bonne séparation entre des séquences voisines ou non dans les études sur l'évolution moléculaire.

On montre que le graphe des points de rupture $G(\gamma_n)$ a au plus deux cycles alternés disjoints. Le sous-graphe de $G(\gamma_n)$ formé par les sommets $\{4, 5, \ldots, n-5, n-4\}$ a une structure régulière (figure 10.8). On oriente toutes les arêtes noires d'un cycle arbitraire de ce sous-graphe de bas en haut et toutes les arêtes grises de haut en bas. Avec cette orientation, toutes les arêtes sont dirigées soit $\nwarrow$, soit $\swarrow$, soit $\downarrow$; par conséquent, avancer le long des arêtes de ce

cycle nous amène lentement mais sûrement vers la gauche. Comment retourner au sommet initial ? On ne peut y revenir qu'après avoir atteint un des sommets « irréguliers » (1 et 3), qui servent de pivots. Le lemme suivant justifie cet argument heuristique.

Lemme 10.1 *Tout cycle alterné dans $G(\gamma_n)$ contient le sommet* 1 *ou le sommet* 3.

Preuve Soit i le sommet impair minimal d'un cycle alterné X dans $G(\gamma_n)$. On considère la suite i, j, k de sommets consécutifs dans X, où (i, j) est noire et (j, k) grise. Si l'on suppose que i est strictement supérieur à 5, on a $j = i - 3$ ou $j = i - 5$ et $k = j + 1$ ou $k = j - 1$ (figure 10.8), ce qui implique que k est impair et qu'il est strictement inférieur à i, d'où une contradiction. Si l'on suppose que i vaut 5, alors j est égal à 2 et k est soit 1, soit 3, une contradiction. Par conséquent, i vaut soit 1, soit 3. ■



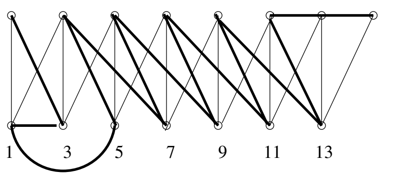


Figure 10.8 – $G(\gamma_{12})$ et $G(\gamma_{13})$.

Théorème 10.3 *(Conjecture de Gollan) Pour tout n, on a : $d(\gamma_n) = d(\gamma_n^{-1}) = n - 1$.*

Preuve Pour $n \leq 2$, l'égalité est triviale. Pour $n > 2$, on partitionne l'ensemble des sommets de $G(\gamma_n)$ en $V_g = \{0, 1, 3\}$ et V_d. D'après le lemme 10.1 et comme V_g ne contient aucun cycle, on voit que tout cycle alterné doit contenir au moins deux arêtes de la coupure (V_g, V_d). Comme celle-ci est constituée de quatre arêtes $((1, 2), (1, 5), (3, 2)$ et $(3, 4))$, le nombre maximal de cycles alternés à arêtes disjointes dans une décomposition cyclique de $G(\gamma_n)$ est d'au plus $\frac{4}{2} = 2$.

D'après le théorème 10.2, on a : $d(\gamma_n) \geq n + 1 - c(\gamma_n) \geq n - 1$. D'autre part, on sait que $d(\gamma_n) \leq n - 1$ car il existe un algorithme simple qui trie chaque permutation de n éléments en $n - 1$ étapes. Enfin, on vérifie l'égalité : $d(\gamma_n^{-1}) = d(\gamma_n)$. ■

Bafna et Pevzner, 1996 [19] ont également prouvé que γ_n et γ_n^{-1} sont les seules permutations dans S_n ayant une distance d'inversion de $n - 1$.

Théorème 10.4 *(Conjecture forte de Gollan) Pour tout n, γ_n et γ_n^{-1} sont les seules permutations qui nécessitent $n - 1$ inversions pour être triées.*

10.4 Espérance de la distance d'inversion

Pour toute permutation $\pi \in S_n$, on considère un ensemble de cycles qui forment une décomposition maximale et on partitionne cet ensemble selon la longueur des cycles. Soit $c_i(\pi)$ le nombre de cycles alternés de longueur i dans une décomposition maximale, qui n'incluent ni le sommet 0, ni le sommet $n+1$. Soit $\delta \leq 2$ le nombre de cycles alternés dans une décomposition maximale qui inclut un de ces deux sommets. On a alors :

$$c(\pi) = \sum_{i=2}^{2(n+1)} c_i(\pi) + \delta. \tag{10.2}$$

Pour $k \leq 2(n+1)$, on considère dans la décomposition les cycles dont la taille est au moins égale à k ; il en existe $c(\pi) - \sum_{i=2}^{k-1} c_i(\pi) - \delta$. Désormais, le graphe des points de rupture de π possède exactement $2(n+1)$ arêtes. Ainsi, les cycles étant à arêtes disjointes, on obtient :

$$\forall k \leq 2(n+1), \qquad c(\pi) - \sum_{i=2}^{k-1} c_i(\pi) - \delta \leq \frac{1}{k}\left(2(n+1) - \sum_{i=2}^{k-1} i c_i(\pi)\right) \tag{10.3}$$

et :

$$\forall k \leq 2(n+1), \qquad c(\pi) \leq \frac{1}{k}\left(2(n+1) + \sum_{i=2}^{k-1} (k-i) c_i(\pi)\right) + \delta. \tag{10.4}$$

Le théorème 10.2, l'inégalité (10.4) et $\delta \leq 2$ impliquent que, pour tout $k \leq 2(n+1)$, on peut minorer $d(\pi)$ de la façon suivante :

$$d(\pi) \geq \left(1 - \frac{2}{k}\right)(n+1) - \frac{1}{k}\left(\sum_{i=2}^{k-1} (k-i) c_i(\pi)\right) - 2 \tag{10.5}$$

$$\geq \left(1 - \frac{2}{k}\right)(n+1) - \left(\sum_{i=2}^{k-1} c_i(\pi)\right) - 2. \tag{10.6}$$

On considère une permutation π choisie uniformément aléatoirement. On note $E(c_i(\pi)) = \frac{1}{n!}\sum_{\pi \in S_n} c_i(\pi)$ le nombre de cycles de longueur i attendu dans une décomposition cyclique maximale de $G(\pi)$. Si l'on peut trouver une borne pour $E(c_i(\pi))$, on peut utiliser (10.6) pour obtenir une borne inférieure pour l'espérance de la distance d'inversion. Le lemme 10.2 fournit une telle borne qui est, de façon assez surprenante, indépendante de n. Notons qu'il y a une légère ambiguïté dans la définition de $E(c_i(\pi))$, qui dépend du choix d'une décomposition cyclique maximale pour chaque $\pi \in S_n$. Cependant, ceci n'affecte en rien le lemme 10.2, qui est vrai pour une décomposition cyclique arbitraire.

Lemme 10.2 *On a :* $E(c_i(\pi)) \leq \frac{2^i}{i}$.

Preuve Un cycle de longueur $i = 2t$ contient t arêtes noires (paires non ordonnées de sommets) de la forme

$$\{(x'_t, x_1), (x'_1, x_2), (x'_2, x_3), \ldots, (x'_{t-1}, x_t)\}, \text{ avec } x_j \sim x'_j.$$

Considérons l'ensemble $x_1, x_2, \ldots, x_t$. Tout d'abord, on affirme que, dans chaque décomposition cyclique maximale, $x_1, x_2, \ldots, x_t$ sont tous distincts. Pour le constater, on considère le cas $x_k = x_l$, pour $1 \leq k < l \leq t$. Alors (x'_k, x_{k+1}), $(x'_{k+1}, x_{k+2}), \ldots, (x'_{l-1}, x_l = x_k)$ forment un cycle alterné, qui peut être détaché pour donner une décomposition plus grande.

On a $\frac{n!}{(n-t)!}$ façons de choisir l'ensemble ordonné $x_1, x_2, \ldots, x_t$. Une fois celui-ci fixé, on a le choix entre au plus deux éléments pour chacun des x'_j, ce qui nous donne une borne de $2^t \frac{n!}{(n-t)!}$ pour le nombre de cycles de longueur $2t$. Notons cependant que l'on compte chaque $(2t)$-cycle $2t$ fois ; $\frac{2^t}{2t} \frac{n!}{(n-t)!}$ est donc une borne mieux ajustée pour le nombre de cycles de longueur $2t$.

On choisit un $(2t)$-cycle arbitraire. Le nombre de permutations dans lesquelles ce cycle peut apparaître ne dépasse pas le nombre de façons de permuter les $n - 2t$ éléments restants plus les t paires qui forment le cycle. De surcroît, chaque paire peut être retournée pour donner un ordre différent, ce qui fournit au plus $2^t(n-t)!$ permutations. Soit p la probabilité qu'un $(2t)$-cycle arbitraire soit présent dans une permutation aléatoire. On a alors $p \leq \frac{2^t(n-t)!}{n!}$ et :

$$\begin{aligned} E(c_i(\pi)) &= E(c_{2t}(\pi)) \\ &\leq \sum_{\{\text{les } (2t)-\text{cycles}\}} p \\ &\leq \frac{2^{2t}}{2t} = \frac{2^i}{i}. \end{aligned}$$

■

Les cycles de longueur 2 correspondent à des adjacences dans la permutation π. Au total, il y a $2n$ adjacences ordonnées. Toute paire de ce type apparaît dans exactement $(n-1)!$ permutations ; la probabilité qu'elle se trouve dans une permutation aléatoire est donc de $\frac{1}{n}$. L'espérance du nombre d'adjacences est donc de $\frac{2n}{n}$, autrement dit $E(c_2) = 2$. Notons que le nombre de points de rupture attendus dans une permutation aléatoire est de $n - 1$.

On utilise le lemme 10.2 et l'égalité $E(c_2) = 2$ pour obtenir une borne inférieure pour l'espérance du diamètre d'inversion :

Théorème 10.5 *(Bafna et Pevzner, 1996 [19]) On a :*

$$E(d(\pi)) \geq \left(1 - \frac{4,5}{\log n}\right) n.$$

Preuve Pour tout $k \leq 2(n+1)$, on déduit de l'inégalité (10.6) :

$$E(d(\pi)) \geq \left(1 - \frac{2}{k}\right)(n+1) - \sum_{i=2}^{k-1} E(c_i) - 2 \geq \left(1 - \frac{2}{k}\right)(n+1) - \sum_{i=4}^{k-1} 2^i/i - 4 \geq$$

$$\left(1 - \frac{2}{k}\right)(n+1) - \sum_{i=4}^{k-1} 2^i + 2^4 - \frac{2^4}{4} - 4 \geq n - \frac{2n}{k} - \left(1 - \frac{2}{k}\right) - 2^k + 1 + 8 \geq$$

$$n - \frac{2n}{k} - 2^k.$$

On choisit $k = \log \frac{n}{\log n}$; on a alors $2^k \leq \frac{n}{k}$ et

$$E(d(\pi)) \geq \left(1 - \frac{3}{\log \frac{n}{\log n}}\right) n \geq \left(1 - \frac{4,5}{\log n}\right) n \quad \text{pour } n \geq 2^{16}.$$

Le lemme 10.2 et l'inégalité (10.5) pour $k = 10$ impliquent que, pour tout n vérifiant $19 < n < 2^{16}$, on a $E(d(\pi)) \geq \left(1 - \frac{4,5}{\log n}\right) n$. Pour $1 \leq n \leq 19$, on obtient : $\left(1 - \frac{4,5}{\log n}\right) n < 1$. ■

10.5 Permutations signées

Soit $\vec{\pi}$ une permutation *signée* de $\{1, \ldots, n\}$, c'est-à-dire une permutation avec un signe + ou un signe − associé à chaque élément. On définit une transformation d'une permutation signée $\vec{\pi}$ d'ordre n en une permutation (non signée) π de $\{1, \ldots, 2n\}$ de la façon suivante. Pour modéliser les signes des éléments dans $\vec{\pi}$, on remplace les éléments positifs $+x$ par $2x - 1, 2x$ et les éléments négatifs $-x$ par $2x, 2x - 1$ (figure 10.9c). La permutation non signée π est appelée l'*image* de la permutation signée $\vec{\pi}$. Dans le graphe des points de rupture $G(\pi)$, les éléments $2x - 1$ et $2x$ sont reliés par une arête noire et une grise pour $1 \leq x \leq n$. Chacune de ces paires constituées d'une arête noire et d'une grise définit un cycle de longueur 2 dans le graphe des points de rupture. Il est clair qu'il existe une décomposition cyclique maximale de $G(\pi)$ qui contient tous ces n cycles de longueur 2. Par définition, le graphe des points de rupture $G(\vec{\pi})$ d'une permutation signée $\vec{\pi}$ est le graphe des points de rupture $G(\pi)$, auquel on enlève ces $2n$ arêtes. On observe que, dans $G(\vec{\pi})$, chaque sommet est de degré 2 (figure 10.9c) et que, par conséquent, le graphe des points de rupture d'une permutation signée est une collection de cycles disjoints. On note $c(\vec{\pi})$ le nombre de tels cycles. On observe que la permutation signée identité d'ordre n se transforme en la permutation identité (non signée) d'ordre $2n$ et que l'effet d'une inversion sur $\vec{\pi}$ entraîne une inversion sur π, impliquant donc $d(\vec{\pi}) \geq d(\pi)$.

Dorénavant, par un tri de l'image $\pi = \pi_1\pi_2 \dots \pi_{2n}$ d'une permutation signée $\vec{\pi}$, on désigne un tri par inversions $\rho(2i+1, 2j)$ de π qui ne « coupe » qu'après les positions paires de π (entre π_{2k-1} et π_{2k} pour $1 \leq k \leq n$). L'effet d'une inversion $\rho(2i+1, 2j)$ sur π peut être simulé par une inversion $\rho(i+1, j)$ sur $\vec{\pi}$, impliquant ainsi l'égalité $d(\vec{\pi}) = d(\pi)$ si les coupures entre π_{2i-1} et π_{2i} sont interdites. Dorénavant, toutes les permutations non signées considérées sont les images de permutations signées. Pour plus de commodité, on étend le terme « permutation signée » aux permutations non signées $\pi = (\pi_1\pi_2 \dots \pi_{2n})$ telles que π_{2i-1} et π_{2i} sont des nombres consécutifs pour $1 \leq i \leq n$. Une inversion $\rho(i, j)$ sur π est dite *légale* si i est impair et j pair. Notons que toute inversion sur une permutation signée correspond à une inversion légale sur son image, et vice versa. Dans la suite, les inversions seront des inversions légales.

Étant donnée une inversion arbitraire ρ, on note $\Delta c \equiv \Delta c(\pi, \rho) = c(\pi \cdot \rho) - c(\pi)$ (augmentation de la taille de la décomposition cyclique). Le théorème 10.1 implique que, pour toute permutation π et toute inversion ρ, on a : $\Delta c \equiv \Delta c(\pi, \rho) \leq 1$. On dit d'une inversion qu'elle est *propre* si elle vérifie $\Delta c = 1$.

Si l'on était capable de trouver une inversion propre pour chaque permutation, on pourrait trier de façon optimale une permutation π en $n + 1 - c(\pi)$ étapes. Cependant, pour une permutation $\pi = (+3, +2, +1)$, il n'y a pas d'inversion propre et, par conséquent, elle ne peut être triée en $n+1-c(\pi) = 2$ étapes (le tri optimal de cette permutation est décrit en figure 10.10). On constate qu'il y a un autre obstacle au tri par inversions. La permutation $\pi = +3+2+1$ comporte un *obstacle* caché pour la trier par inversions. La notion d'obstacle sera définie dans la section suivante.

On dit qu'une inversion $\rho(i, j)$ *agit* sur les arêtes noires (π_{i-1}, π_i) et (π_j, π_{j+1}) dans $G(\pi)$. L'inversion $\rho(i, j)$ est une *inversion (qui agit) sur un cycle* C de $G(\pi)$ si les arêtes noires (π_{i-1}, π_i) et (π_j, π_{j+1}) appartiennent à C. Une arête grise g est dite *orientée* si une inversion qui agit sur deux arêtes noires incidentes à g est propre ; elle est dite *non orientée* sinon. Par exemple, les arêtes grises $(8, 9)$ et $(22, 23)$ de la figure 10.9c sont orientées, alors que les arêtes grises $(4, 5)$ et $(18, 19)$ ne le sont pas.

Lemme 10.3 *Soit* (π_i, π_j) *une arête grise incidente aux arêtes noires* (π_k, π_i) *et* (π_j, π_l). *Alors* (π_i, π_j) *est orientée si et seulement si* $i - k = j - l$.

Dans $G(\pi)$, un cycle est dit *orienté* s'il possède une arête grise orientée ; sinon, il est dit non orienté. Les cycles C et F dans la figure 10.9c sont orientés, alors que les cycles A, B, D et E sont non orientés. Il est clair qu'il n'existe pas d'inversion propre agissant sur un cycle non orienté.

10.6 Graphes de chevauchements et obstacles

Dans $G(\pi)$, les arêtes grises (π_i, π_j) et (π_k, π_t) sont dites *chevauchantes* si les intervalles $[i, j]$ et $[k, t]$ se chevauchent et qu'aucun des deux ne contient l'autre. Par exemple, les arêtes $(4, 5)$ et $(18, 19)$ dans la figure 10.9c sont chevauchantes,

tandis que les arêtes $(4, 5)$ et $(22, 23)$, tout comme $(4, 5)$ et $(16, 17)$, sont non-chevauchantes. Les cycles C_1 et C_2 sont *chevauchants* s'il existe des arêtes grises chevauchantes $g_1 \in C_1$ et $g_2 \in C_2$.

Soit $\mathcal{C}_\pi$ l'ensemble des cycles dans le graphe des points de rupture d'une permutation π. On définit le graphe de *chevauchements* $H_\pi(\mathcal{C}_\pi, \mathcal{I}_\pi)$ de π comme l'ensemble d'arêtes :

$$\mathcal{I}_\pi = \{(C_1, C_2) :\ C_1 \text{ et } C_2 \text{ sont des cycles chevauchants dans } G(\pi)\}.$$

La figure 10.9d montre un graphe de chevauchements H_π constitué de trois composantes connexes. L'ensemble des sommets de H_π est partitionné en sommets *orientés* et *non orientés* (cycles dans $\mathcal{C}_\pi$). Une composante connexe de H_π est dite *orientée* si elle possède au moins un sommet orienté ; sinon, elle est dite *non orientée*. Pour une composante connexe U, on définit les positions à l'extrême gauche et à l'extrême droite de U comme étant

$$U_{min} = \min_{\pi_i \in C \in U} i \ \text{ et } \ U_{max} = \max_{\pi_i \in C \in U} i.$$

Par exemple, la composante U qui contient les cycles B, C et D dans la figure 10.9c a $\pi_2 = 6$ pour sommet extrême gauche et $\pi_{13} = 17$ comme extrême droit ; par conséquent, on a : $[U_{min}, U_{max}] = [2, 13]$.

On dit qu'une composante U *sépare* les composantes U' et U'' dans π s'il existe une arête grise (π_i, π_j) dans U telle que l'on ait $[U'_{min}, U'_{max}] \subset [i, j]$ et $[U''_{min}, U''_{max}] \not\subset [i, j]$. Par exemple, la composante U en figure 10.11a sépare les composantes U' et U''.

Soit $\prec$ un ordre partiel sur un ensemble P. Un élément $x \in P$ est appelé un élément *minimal* pour $\prec$ s'il n'existe pas d'élément $y \in P$ tel que $y \prec x$. Un élément $x \in P$ est le *plus grand* pour $\prec$ s'il vérifie $y \prec x$ pour tout $y \in P$.

Considérons l'ensemble des composantes non orientées $\mathcal{U}_\pi$ dans H_π et définissons l'ordre partiel de *confinement* sur cet ensemble, i.e. $U \prec W$ si l'on a $[U_{min}, U_{max}] \subset [W_{min}, W_{max}]$ pour $(U, W) \in \mathcal{U}_\pi^2$. Par définition, un *obstacle* est une composante non orientée qui est soit un obstacle minimal, soit le plus grand obstacle (un *obstacle minimal* $U \in \mathcal{U}_\pi$ est un élément minimal pour $\prec$ et *le plus grand obstacle* vérifie les deux conditions suivantes : (i) U est le plus grand élément pour $\prec$ *et* (ii) U ne sépare pas deux obstacles). Soit $h(\pi)$ le nombre *global* d'obstacles dans π. La permutation π dans la figure 10.9c possède une composante non orientée et $h(\pi)$ vaut 1. La permutation π dans la figure 10.11b a deux obstacles minimaux et un plus grand ($h(\pi) = 3$). La permutation π dans la figure 10.11a présente deux obstacles minimaux mais pas de plus grand obstacle ($h(\pi) = 2$), car la plus grande composante non orientée U dans la figure 10.11a sépare U' et U''.

Le théorème suivant améliore encore la borne pour le tri par inversions des permutations signées.

Théorème 10.6 *Pour une permutation (signée) arbitraire π, on a : $d(\pi) \geq n + 1 - c(\pi) + h(\pi)$.*

(a)

3	5	8	6	4	7	9	2	1	10	11
3	5	4	6	8	7	9	2	1	10	11
3	4	5	6	8	7	9	2	1	10	11
3	4	5	6	7	8	9	2	1	10	11
9	8	7	6	5	4	3	2	1	10	11
1	2	3	4	5	6	7	8	9	10	11

(b)

arête noire
arête grise

0 3 5 8 6 4 7 9 2 1 10 11 12

(c)

A B C D E F

0 5 6 10 9 15 16 12 11 7 8 14 13 17 18 3 4 1 2 19 20 22 21 23

+3 -5 +8 -6 +4 -7 +9 +2 +1 +10 -11

(d)

B D C A E F

cycle non orienté
cycle orienté

Figure 10.9 – (a) Tri optimal d'une permutation (3 5 8 6 4 7 9 2 1 10 11) par cinq inversions et (b) graphe des points de rupture de cette permutation ; (c) transformation d'une permutation signée en une permutation non signée π et graphe des points de rupture $G(\pi)$; (d) graphe de chevauchements H_π avec deux composantes orientées et une composante non orientée.

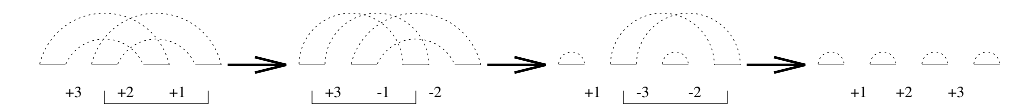


Figure 10.10 – Le tri optimal de la permutation $\pi = +3 +2 +1$ implique une inversion impropre.

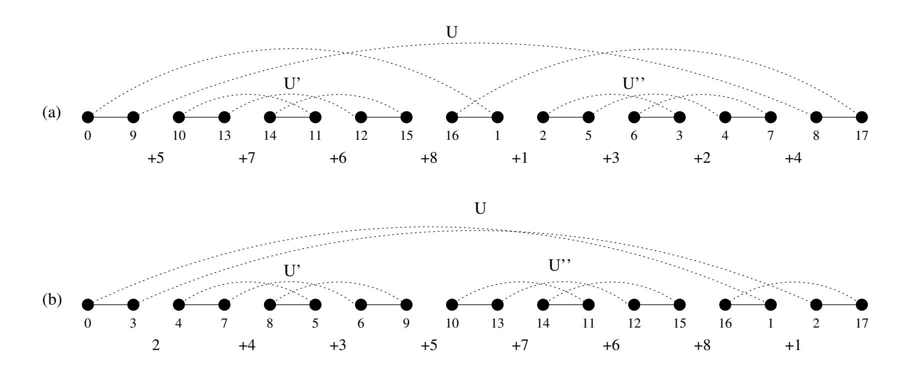


Figure 10.11 – (a) La composante non orientée U sépare U' et U'' avec l'arête $(0,1)$; (b) l'obstacle U ne sépare pas U' et U''.

Preuve Étant donnée une inversion arbitraire ρ, on note $\Delta h \equiv \Delta h(\pi,\rho) = h(\pi \cdot \rho) - h(\pi)$. Il est clair que toute inversion ρ agit sur les arêtes noires d'au plus deux obstacles ; par conséquent, ρ peut « détruire » au plus deux obstacles minimaux. Notons que, si ρ détruit deux obstacles minimaux dans $\mathcal{U}_\pi$, alors ρ ne peut détruire le plus grand obstacle dans $\mathcal{U}_\pi$ (voir la condition (ii) dans la définition du plus grand obstacle). On a donc $\Delta h \geq -2$ pour toute inversion ρ.

Le théorème 10.1 implique $\Delta c \in \{-1, 0, 1\}$. Si Δc vaut 1, ρ agit sur un cycle orienté et n'affecte donc aucun obstacle dans π. Par conséquent, on a $\Delta h = 0$ et $\Delta(c-h) \equiv \Delta c - \Delta h = 1$. Si Δc vaut 0, ρ agit sur un cycle et il affecte donc au plus un obstacle (voir la condition (ii) dans la définition du plus grand obstacle). Ceci implique les inégalités $\Delta h \geq -1$ et $\Delta(c-h) \leq 1$. Si Δc vaut -1, on obtient $\Delta(c-h) \leq 1$, d'après l'inégalité $\Delta h \geq -2$ pour toute inversion ρ.

Par conséquent, pour une inversion arbitraire ρ, on a : $\Delta(c-h) \leq 1$. Si l'on tient compte de $c(\iota) = n+1$ et $h(\iota) = 0$ pour la permutation identité ι, on en déduit : $d(\pi) \geq (c(\iota) - h(\iota)) - (c(\pi) - h(\pi)) = n + 1 - c(\pi) + h(\pi)$. ■

Hannenhalli et Pevzner, 1995 [154] ont prouvé que la borne inférieure $d(\pi) \geq n + 1 - c(\pi) + h(\pi)$ est très bien ajustée. Comme premier pas vers la borne supérieure $d(\pi) \leq n+1-c(\pi)+h(\pi)+1$, nous avons développé une technique appelée *transformations équivalentes* de permutations.

10.7 Transformations équivalentes de permutations

La structure compliquée du chevauchement des longs cycles dans les graphes des points de rupture pose de sérieux problèmes pour l'analyse du tri par inversions. Pour les contourner, on introduit les transformations équivalentes de permutations, qui sont fondées sur l'idée suivante : si une permutation $\pi \equiv \pi(0)$ possède un cycle long, on la transforme en une nouvelle permutation $\pi(1)$ en

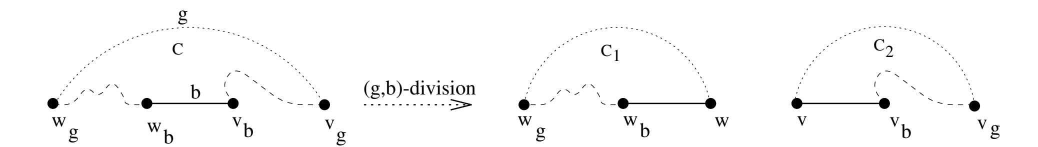


Figure 10.12 – Exemple de (g, b)-division.

le « cassant » en deux cycles plus petits. On continue avec $\pi(1)$ de la même manière et on forme une suite de permutations $\pi \equiv \pi(0), \pi(1), \ldots, \pi(k) \equiv \sigma$, qui se termine avec une permutation *simple* (c'est-à-dire ne possédant pas de cycle long). Dans cette partie, on montre que ces transformations peuvent être arrangées de sorte que chaque tri de σ imite un tri de π avec le même nombre d'inversions. Les parties suivantes montrent comment trier de façon optimale des permutations simples. Le tri optimal de la permutation *simple* σ imite le tri optimal de la permutation *arbitraire* π, aboutissant à un algorithme polynomial pour le tri par inversions.

Soit $b = (v_b, w_b)$ une arête noire et $g = (w_g, v_g)$ une arête grise appartenant à un cycle $C = \ldots, v_b, w_b, \ldots, w_g, v_g, \ldots$ dans le graphe de points de rupture $G(\pi)$ d'une permutation π. Une (g, b)-*division* de $G(\pi)$ est un nouveau graphe $\hat{G}(\pi)$ obtenu à partir de $G(\pi)$ en

– enlevant les arêtes g et b,

– ajoutant deux nouveaux sommets v et w,

– ajoutant deux nouvelles arêtes noires (v_b, v) et (w, w_b),

– ajoutant deux nouvelles arêtes grises (w_g, w) et (v, v_g).

La figure 10.12 montre une (g, b)-division transformant un cycle C dans $G(\pi)$ en deux cycles C_1 et C_2 dans $\hat{G}(\pi)$. Si $G(\pi)$ est un graphe de points de rupture d'une permutation signée π, alors toute (g, b)-division de $G(\pi)$ correspond à un graphe de points de rupture d'une permutation signée *généralisée* $\hat{\pi}$ qui vérifie $\hat{G}(\pi) = G(\hat{\pi})$. Ci-dessous, on définit les permutations généralisées et on décrit la procédure de *remplissage* pour trouver une permutation généralisée $\hat{\pi}$ correspondant à une (g, b)-division de G.

Une permutation généralisée $\pi = \pi_1\pi_2 \ldots \pi_n$ est une permutation de *réels* arbitraires distincts (au lieu d'une permutation d'*entiers* $\{1, 2, \ldots, n\}$). Dans cette section, lorsque nous évoquerons une permutation, il s'agira d'une permutation généralisée ; la *permutation identité généralisée* désignera une permutation généralisée $\pi = \pi_1\pi_2 \ldots \pi_n$, avec $\pi_i < \pi_{i+1}$ pour $1 \leq i \leq n - 1$. On étend une permutation $\pi = \pi_1\pi_2 \ldots \pi_n$ en ajoutant $\pi_0 = \min_{1 \leq i \leq n} \pi_i - 1$ et $\pi_{n+1} = \max_{1 \leq i \leq n} \pi_i + 1$. Les éléments π_j et π_k de π sont dits *consécutifs* s'il n'existe pas d'élément π_l vérifiant $\pi_j < \pi_l < \pi_k$, pour $1 \leq l \leq n$. Les éléments

π_i et π_{i+1} de π sont dits *adjacents* pour $0 \leq i \leq n$. Par définition, le *graphe de points de rupture* d'une permutation (généralisée) $\pi = \pi_1\pi_2 \ldots \pi_n$ est le graphe d'ensemble de sommets $\{\pi_0, \pi_1, \ldots, \pi_n, \pi_{n+1}\}$, avec des arêtes noires entre les éléments adjacents non consécutifs et des arêtes grises entre des éléments consécutifs non adjacents. Évidemment, la définition du graphe de points de rupture pour les permutations généralisées est cohérente avec la notion de graphe de points de rupture décrite précédemment.

Soit $b = (\pi_{i+1}, \pi_i)$ une arête noire et $g = (\pi_j, \pi_k)$ une arête grise appartenant à un cycle $C = \ldots, \pi_{i+1}, \pi_i, \ldots, \pi_j, \pi_k, \ldots$ dans le graphe de points de rupture $G(\pi)$. On définit $\Delta = \pi_k - \pi_j$ et on pose $v = \pi_j + \frac{\Delta}{3}$ et $w = \pi_k - \frac{\Delta}{3}$. Un (g,b)-*remplissage* de $\pi = (\pi_1\pi_2 \ldots \pi_n)$ est une permutation sur $n{+}2$ éléments obtenue à partir de π en insérant v et w après le i-ième élément de π $(0 \leq i \leq n)$:

$$\hat{\pi} = \pi_1\pi_2 \ldots \pi_i v w \pi_{i+1} \ldots \pi_n.$$

On note que v et w sont à la fois consécutifs et adjacents dans $\hat{\pi}$, ce qui implique donc que, si π est (l'image d') une permutation signée, alors $\hat{\pi}$ est également (l'image d') une permutation signée. Le lemme suivant établit la correspondance entre les (g,b)-remplissages et les (g,b)-divisions† .

Lemme 10.4 $\hat{G}(\pi) = G(\hat{\pi})$.

Si g et b sont des arêtes non-incidentes d'un *long* cycle C dans $G(\pi)$, alors le (g,b)-remplissage casse C en deux cycles *plus petits* dans $G(\hat{\pi})$. Par conséquent, les remplissages peuvent être utilisés pour transformer une permutation arbitraire π en une permutation simple. Notons que le nombre d'éléments dans $\hat{\pi}$ est $\hat{n} = n + 1$ et que l'on a $c(\hat{\pi}) = c(\pi) + 1$. Ci-dessous, on prouve que, pour toute permutation ayant un cycle long, il existe un remplissage d'arêtes non-incidentes de celui-ci qui vérifie $h(\hat{\pi}) = h(\pi)$, indiquant ainsi que le remplissage donne une façon d'éliminer les longs cycles dans une permutation sans changer le paramètre $n + 1 - c(\pi) + h(\pi)$. Tout d'abord, on a besoin d'une série de lemmes techniques.

Lemme 10.5 *Supposons qu'un (g,b)-remplissage sur un cycle C dans $G(\pi)$ supprime l'arête grise g et ajoute deux nouvelles arêtes grises g_1 et g_2. Si g est orientée, alors soit g_1 soit g_2 est orientée dans $G(\hat{\pi})$. Si C est non orientée, alors g_1 et g_2 sont non orientées dans $G(\hat{\pi})$.*

Lemme 10.6 *Supposons qu'un (g,b)-remplissage scinde un cycle C dans $G(\pi)$ en deux cycles C_1 et C_2 dans $G(\hat{\pi})$. Alors C est orienté si et seulement si soit C_1 soit C_2 est orienté.*

---

†Évidemment, un (g,b)-remplissage d'une permutation $\pi = (\pi_1\pi_2 \ldots \pi_n)$ sur $\{1, 2, \ldots, n\}$ peut être modélisé par une permutation $\hat{\pi} = (\hat{\pi}_1\hat{\pi}_2 \ldots \hat{\pi}_i v w \hat{\pi}_{i+1} \ldots \hat{\pi}_n)$ sur $\{1, 2, \ldots, n+2\}$, avec $v = \pi_j + 1$, $w = \pi_k + 1$ et $\hat{\pi}_i = \pi_i + 2$ si π_i est strictement supérieur à $\min\{\pi_j, \pi_k\}$ et $\hat{\pi}_i = \pi_i$ sinon. Les permutations généralisées ont été introduites pour rendre la procédure « d'imitation » qui suit plus intuitive.

Preuve Notons qu'un (g, b)-remplissage préserve l'orientation des arêtes grises de $G(\hat{\pi})$ qui sont « héritées » de $G(\pi)$ (lemme 10.3). Si C est orienté, il possède obligatoirement une arête grise orientée. Si cette arête est différente de g, alors elle reste orientée dans un (g, b)-remplissage de π et, par conséquent, un cycle (C_1 ou C_2) contenant cette arête est orienté. Si $g = (w_g, v_g)$ est la seule arête grise orientée dans C, alors un (g, b)-remplissage ajoute deux nouvelles arêtes grises ((w_g, w) et (v, v_g)) à $G(\hat{\pi})$, dont l'une est orientée (lemme 10.5). Un cycle (C_1 ou C_2) qui contient cette arête est donc orienté.

Si C est un cycle non orienté, alors toutes les arêtes de C_1 et C_2 héritées de C demeurent non orientées. Le lemme 10.5 implique que les nouvelles arêtes ((w_g, w) et (v, v_g)) dans C_1 et C_2 sont également non orientées. ■

Le lemme suivant établit que les remplissages préservent le chevauchement des arêtes grises.

Lemme 10.7 *Soient g' et g'' deux arêtes grises de $G(\pi)$ différentes de g. Alors g' et g'' sont chevauchantes dans π si et seulement si g' et g'' sont chevauchantes dans un (g, b)-remplissage de π.*

Ce lemme implique immédiatement le suivant :

Lemme 10.8 *Supposons qu'un (g, b)-remplissage scinde un cycle C dans $G(\pi)$ en deux cycles C_1 et C_2 dans $G(\hat{\pi})$. Alors tout cycle D chevauchant C dans $G(\pi)$ chevauche soit C_1 soit C_2 dans $G(\hat{\pi})$.*

Preuve Soient $d \in D$ et $c \in C$ deux arêtes grises chevauchantes dans $G(\pi)$. Si c est différente de g, alors le lemme 10.7 implique que d et c sont chevauchantes dans $G(\hat{\pi})$; par conséquent, D chevauche soit C_1 soit C_2. Si c et g sont confondues, il est facile de voir que l'une des nouvelles arêtes grises dans $G(\hat{\pi})$ chevauche d ; D chevauche donc C_1 ou C_2 dans $G(\hat{\pi})$. ■

Lemme 10.9 *Pour toute arête grise g, il existe une arête grise f chevauchant g dans $G(\pi)$.*

Lemme 10.10 *Soient C un cycle dans $G(\pi)$ et $g \notin C$ une arête grise dans $G(\pi)$. Alors g chevauche un nombre pair d'arêtes grises dans C.*

Un (g, b)-remplissage ϕ transformant π en $\hat{\pi}$ (c'est-à-dire $\hat{\pi} = \pi \cdot \phi$) est dit *solide* s'il agit sur des arêtes non-incidentes d'un cycle long et si $h(\pi)$ est égal à $h(\hat{\pi})$. Il est clair que tout remplissage solide scinde un cycle long en deux cycles plus petits.

Théorème 10.7 *Si C est un cycle long dans $G(\pi)$, alors il existe un (g, b)-remplissage solide qui agit sur C.*

Preuve Si C possède une paire d'arêtes grises chevauchantes $(g_1, g_2) \in C^2$, le fait de retirer ces arêtes transforme C en deux chemins. Comme C est un cycle long, au moins l'un de ces chemins contient une arête grise g. On choisit une arête noire b d'un autre chemin et l'on considère le (g, b)-remplissage qui transforme π en $\hat{\pi}$ (il est clair que g et b sont des arêtes non-incidentes). Ce (g, b)-remplissage scinde C en deux cycles C_1 et C_2 dans $G(\hat{\pi})$, avec g_1 et g_2 appartenant à des cycles différents C_1 et C_2. D'après le lemme 10.7, g_1 et g_2 se chevauchent, ce qui implique que C_1 et C_2 se chevauchent. Ce (g, b)-remplissage ne scinde pas la composante K de H_π qui contient le cycle C car, d'après le lemme 10.8, tous les cycles de K appartiennent à la composante de $H_{\hat{\pi}}$ qui contient C_1 et C_2. De plus, selon le lemme 10.6, l'orientation de cette composante est la même dans H_π et $H_{\hat{\pi}}$. Par conséquent, le (g, b)-remplissage choisi préserve l'ensemble des obstacles et on a $h(\pi) = h(\hat{\pi})$.

Si toutes les arêtes grises de C sont mutuellement non chevauchantes, alors C est un cycle non orienté. Les lemmes 10.9 et 10.10 impliquent qu'il existe une arête grise $e \in C'$ qui chevauche au moins deux arêtes grises g_1 et g_2 de C. Le fait de retirer g_1 et g_2 transforme C en deux chemins et, comme C est un cycle long, au moins l'un de ces chemins contient une arête grise g. On choisit une arête noire b d'un autre chemin et on considère le (g, b)-remplissage de π. Il scinde C en deux cycles C_1 et C_2 dans $G(\hat{\pi})$, avec g_1 et g_2 appartenant à des cycles différents C_1 et C_2. D'après le lemme 10.7, C_1 et C_2 chevauchent C' dans $\hat{\pi}$. Par conséquent, ce (g, b)-remplissage ne scinde pas la composante K de H_π qui contient C et C'. De plus, selon le lemme 10.6, C_1 et C_2 sont non orientés, ce qui implique que l'orientation de cette composante est la même dans H_π et $H_{\hat{\pi}}$. Par conséquent, le (g, b)-remplissage choisi préserve l'ensemble des obstacles, d'où $h(\pi) = h(\hat{\pi})$. ■

Une permutation π est dite *équivalente* à une permutation σ (et on note $\pi \rightsquigarrow \sigma$) s'il existe une série de permutations $\pi \equiv \pi(0), \pi(1), \ldots, \pi(k) \equiv \sigma$ qui vérifient $\pi(i+1) = \pi(i) \cdot \phi(i)$ pour un (g, b)-remplissage solide $\phi(i)$ qui agit sur π_i $(0 \leq i \leq k-1)$.

Théorème 10.8 *Toute permutation est équivalente à une permutation simple.*

Preuve On définit la *complexité* d'une permutation π comme étant la somme $\sum_{C \in \mathcal{C}_\pi} (l(C) - 2)$, où $\mathcal{C}_\pi$ est l'ensemble des cycles dans $G(\pi)$ et $l(C)$ la longueur d'un cycle C. La complexité d'une permutation simple est nulle. Notons que tout remplissage sur des arêtes non-incidentes d'un cycle long C scinde C en deux cycles C_1 et C_2, avec $l(C) = l(C_1) + l(C_2) - 1$. Par conséquent :

$$(l(C) - 2) = (l(C_1) - 2) + (l(C_2) - 2) + 1,$$

ce qui implique qu'un remplissage sur des arêtes non-incidentes d'un cycle réduit la complexité des permutations. Cette observation associée au théorème 10.7 implique que toute permutation avec des cycles longs peut être

transformée en une permutation sans cycle long par une série de remplissages préservant $b(\pi) - c(\pi) + h(\pi)$. ■

Soit $\hat{\pi}$ un (g, b)-remplissage de π et soit ρ une inversion qui agit sur deux arêtes noires de $\hat{\pi}$. Alors ρ peut être simulée sur π en ignorant les éléments remplis. On a besoin d'une généralisation de cette observation. Une suite de permutations généralisées $\pi \equiv \pi(0), \pi(1), \ldots, \pi(k) \equiv \sigma$ est appelée un *tri généralisé* de π si σ est la permutation identité (généralisée) et si $\pi(i+1)$ est obtenue à partir de $\pi(i)$ soit par une inversion, soit par un remplissage. Notons que les inversions et les remplissages dans un tri généralisé de π peuvent se chevaucher.

Lemme 10.11 *Tout tri généralisé de π simule un (véritable) tri de π avec le même nombre d'inversions.*

Preuve On ignore les éléments remplis. ■

Dans la suite, on montre comment trouver un tri généralisé d'une permutation π par une série de remplissages et d'inversions comportant $d(\pi)$ inversions. Le lemme 10.11 implique que ce tri généralisé de π imite un (véritable) tri optimal de π.

10.8 Recherche d'inversions solides

Rappelons que, pour une inversion arbitraire, on a : $\Delta(c - h) \leq 1$ (voir preuve du théorème 10.6). Une inversion ρ est dite *solide* si elle vérifie l'égalité $\Delta(c - h) = 1$. La première inversion de la figure 10.10 n'est pas propre, mais elle est solide ($\Delta c = 0$ et $\Delta h = -1$). La figure 10.14 présente des exemples d'inversions solides ($\Delta c = 1$, $\Delta h = 0$) et non solides ($\Delta c = 1$, $\Delta h = 1$). Par la suite, on prouve l'existence d'une inversion solide qui agit sur un cycle dans une composante orientée en analysant les actions des inversions sur des permutations simples. Dans cette section, lorsque l'on parlera de cycles, il s'agira de cycles *courts* et les permutations seront des permutations simples.

Soit C un cycle dans $G(\pi)$. On note $V(C)$ l'ensemble des sommets de tous les cycles chevauchant C (autrement dit, $V(C)$ est l'ensemble des sommets adjacents à C dans H_π). On définit les ensembles d'arêtes dans le sous-graphe de H_π induit par $V(C)$:

$$E(C) = \{(C_1, C_2) : C_1, C_2 \in V(C) \text{ et } C_1 \text{ chevauche } C_2 \text{ dans } \pi\},$$

et son complémentaire :

$$\overline{E}(C) = \{(C_1, C_2) : C_1, C_2 \in V(C) \text{ et } C_1 \text{ ne chevauche pas } C_2 \text{ dans } \pi\}.$$

Une inversion ρ qui agit sur un cycle (court) orienté C « détruit » C (autrement dit, elle enlève les arêtes de C de $G(\pi)$) et transforme tout autre cycle

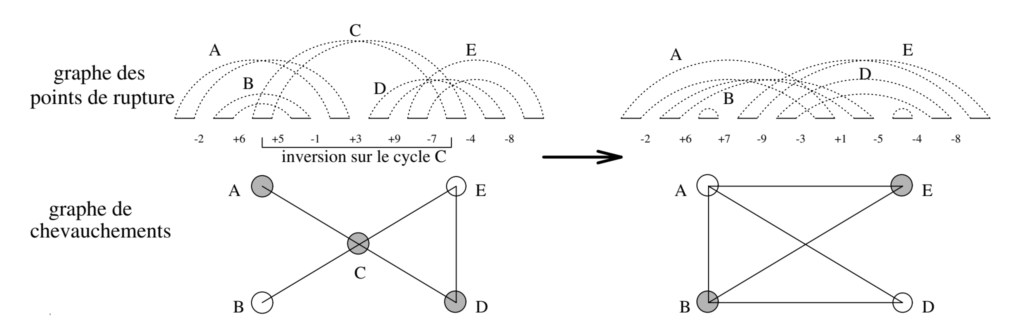


Figure 10.13 – L'inversion sur un cycle C complète les arêtes entre les voisins de C et change l'orientation de chaque cycle voisin de C dans le graphe de chevauchements.

dans $G(\pi)$ en un cycle correspondant sur les mêmes sommets dans $G(\pi \cdot \rho)$. Par suite, ρ transforme le graphe de chevauchements $H_\pi(\mathcal{C}_\pi, \mathcal{I}_\pi)$ de π en le graphe de chevauchements $H_{\pi\cdot\rho}(\mathcal{C}_\pi \setminus C, \mathcal{I}_{\pi\cdot\rho})$ de $\pi \cdot \rho$. Cette transformation a pour effet de compléter le sous-graphe induit par $V(C)$, comme le décrit le lemme suivant (figure 10.13). On note : $\overline{\mathcal{I}}_\pi = \mathcal{I}_\pi \setminus \{(C, D) : \ D \in V(C)\}$.

Lemme 10.12 *Soit ρ une inversion qui agit sur un cycle (court) orienté C. Alors*

- $\mathcal{I}_{\pi\cdot\rho} = (\overline{\mathcal{I}}_\pi \setminus E(C)) \cup \overline{E}(C)$, *i.e. ρ enlève les arêtes de $E(C)$ et ajoute les arêtes de $\overline{E}(C)$ pour transformer H_π en $H_{\pi\cdot\rho}$ et*

- *ρ change l'orientation d'un cycle $D \in \mathcal{C}_\pi$ si et seulement s'il vérifie $D \in V(C)$.*

Le lemme 10.12 implique immédiatement le lemme suivant :

Lemme 10.13 *Soit ρ une inversion qui agit sur un cycle C et soient A et B des sommets non adjacents dans $H_{\pi\cdot\rho}$. Alors (A, B) est une arête dans H_π si et seulement si A et B appartiennent à $V(C)$.*

Soit K une composante orientée de H_π et soit $\mathcal{R}(K)$ un ensemble d'inversions agissant sur les cycles orientés de K. Supposons qu'une inversion $\rho \in \mathcal{R}(K)$ « scinde » K en un certain nombre de composantes connexes $K_1(\rho)$, $K_2(\rho), \ldots$ dans $H_{\pi\cdot\rho}$ et que les m premières composantes soient non orientées. Si m est strictement positif, ρ peut ne pas être solide, car certaines des composantes $K_1(\rho), \ldots, K_m(\rho)$ peuvent former de nouveaux obstacles dans $\pi \cdot \rho$, augmentant ainsi $h(\pi \cdot \rho)$ en comparaison de $h(\pi)$ (figure 10.14). Dans la suite, on montre qu'il y a de la flexibilité dans le choix d'une inversion de l'ensemble $\mathcal{R}(K)$, ce qui permet de substituer une inversion solide σ à une inversion non solide ρ.

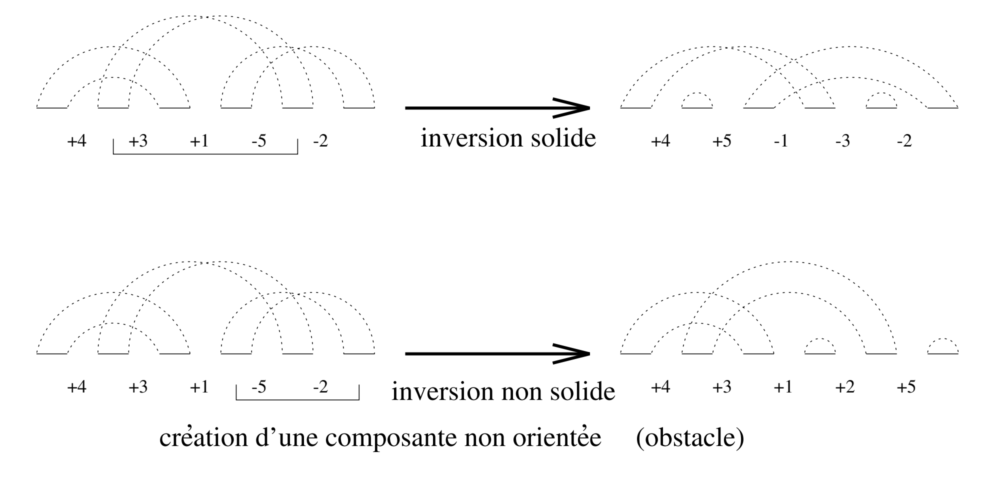


Figure 10.14 – Exemples d'inversions solides et non solides.

Lemme 10.14 *Soient ρ et σ des inversions qui agissent sur deux cycles orientés chevauchants C et C', respectivement, dans $G(\pi)$. Si C' appartient à une composante non orientée $K_1(\rho)$ dans $H_{\pi\cdot\rho}$, alors*

- *toute paire de sommets extérieurs à $K_1(\rho)$ qui sont adjacents dans $H_{\pi\cdot\rho}$ le sont également dans $H_{\pi\sigma}$ et*

- *l'orientation des sommets extérieurs à $K_1(\rho)$ ne change pas dans $H_{\pi\sigma}$ comparé à $H_{\pi\cdot\rho}$.*

Preuve Soient D et E deux sommets à l'extérieur de $K_1(\rho)$ reliés par une arête dans $H_{\pi\cdot\rho}$. Si l'un de ces sommets, disons D, n'appartient pas à $V(C)$ dans H_π, alors le lemme 10.13 implique que (i) (C', D) n'est pas une arête dans H_π et (ii) (D, E) est une arête dans H_π. Par conséquent, d'après le lemme 10.12, l'inversion σ préserve l'arête (D, E) dans $H_{\pi\sigma}$. Si les deux sommets D et E appartiennent à $V(C)$, le lemme 10.12 implique que (D, E) n'est pas une arête dans H_π. Comme le sommet C' et les sommets D et E sont situés dans des composantes différentes de $H_{\pi\cdot\rho}$, le lemme 10.13 implique que (C', D) et (C', E) sont des arêtes dans H_π. Par conséquent, d'après le lemme 10.12, (D, E) est une arête dans $H_{\pi\sigma}$. Dans les deux cas, σ préserve l'arête (D, E) dans $H_{\pi\sigma}$ et la première partie du lemme est vérifiée.

Le lemme 10.13 implique que, pour tout sommet D extérieur à $K_1(\rho)$, D appartient à $V(C)$ si et seulement si D appartient à $V(C')$. Cette observation et le lemme 10.12 impliquent que l'orientation des sommets extérieurs à $K_1(\rho)$ ne change pas dans $H_{\pi\sigma}$ en comparaison de $H_{\pi\cdot\rho}$. ■

Lemme 10.15 *Toute composante non orientée dans le graphe de chevauchements (d'une permutation simple) contient au moins deux sommets.*

Preuve D'après le lemme 10.9, toute arête grise dans $G(\pi)$ est chevauchée par une arête grise. Par conséquent, tout cycle (court) non orienté dans $G(\pi)$ possède un cycle chevauchant. ■

Théorème 10.9 *Pour toute composante orientée K dans H_π, il existe une inversion (solide) $\rho \in \mathcal{R}(K)$ telle que toutes les composantes $K_1(\rho), K_2(\rho), \ldots$ sont orientées dans $H_{\pi \cdot \rho}$.*

Preuve Supposons qu'une inversion $\rho \in \mathcal{R}(K)$ « scinde » K en un certain nombre de composantes connexes $K_1(\rho), K_2(\rho), \ldots$ dans $H_{\pi \cdot \rho}$ et que les m premières soient non orientées. On désigne par $indice(\rho) = \sum_{i=1}^{m} |K_i(\rho)|$ le nombre total de sommets dans ces composantes non orientées, où $|K_i(\rho)|$ représente le nombre de sommets dans $K_i(\rho)$. Soit ρ une inversion qui vérifie :

$$indice(\rho) = \min_{\sigma \in \mathcal{R}(K)} indice(\sigma).$$

Cette inversion agit sur un cycle C et scinde K en plusieurs composantes. Si elles sont toutes orientées (i.e. $indice(\rho) = 0$), le théorème est vrai. Sinon, $indice(\rho)$ est strictement positif et on note $K_1(\rho), \ldots, K_m(\rho)$ $(m \geq 1)$ les composantes non orientées dans $H_{\pi \cdot \rho}$. Ci-dessous, on trouve une autre inversion $\sigma \in \mathcal{R}(K)$ avec $indice(\sigma) < indice(\rho)$, ce qui est contradictoire.

Soit V_1 l'ensemble des sommets de la composante $K_1(\rho)$ dans $H_{\pi \cdot \rho}$. Notons que $K_1(\rho)$ contient au moins un sommet de $V(C)$; on considère l'ensemble (non vide) $V = V_1 \cap V(C)$, formé des sommets de la composante $K_1(\rho)$ adjacents à C dans H_π. Comme $K_1(\rho)$ est une composante non orientée dans $\pi \cdot \rho$, tous les cycles de V sont orientés dans π et tous les cycles de $V_1 \setminus V$ sont non orientés dans π (lemme 10.12). Soit C' un cycle (orienté) dans V et soit σ l'inversion qui agit sur C' dans $G(\pi)$. Le lemme 10.14 implique que, pour tout $i \geq 2$, toutes les arêtes de la composante $K_i(\rho)$ dans $H_{\pi \cdot \rho}$ sont préservées dans $H_{\pi\sigma}$ et que l'orientation des sommets dans $K_i(\rho)$ ne change pas dans $H_{\pi\sigma}$ comparé à $H_{\pi \cdot \rho}$. Par conséquent, toutes les composantes non orientées $K_{m+1}(\rho), K_{m+2}(\rho), \ldots$ de $\pi \cdot \rho$ « survivent » dans $\pi\sigma$ et on a :

$$indice(\sigma) \leq indice(\rho).$$

Ci-dessous, on prouve qu'il existe une inversion σ qui agit sur un cycle de V et qui vérifie $indice(\sigma) < indice(\rho)$, une contradiction.

Si V_1 est différent de $V(C)$, il existe une arête entre un cycle (orienté) $C' \in V$ et un cycle (non orienté) $C'' \in V_1 \setminus V$ dans $G(\pi)$. Le lemme 10.12 implique qu'une inversion σ agissant sur C' dans π oriente le cycle C'' dans $G(\pi)$. Cette observation et le lemme 10.14 impliquent que σ réduit $indice(\sigma)$ d'au moins 1 en comparaison de $indice(\rho)$, ce qui est contradictoire.

Si l'on a $V_1 = V(C)$ (tous les cycles de K_1 chevauchent C), il existe au moins deux sommets dans $V(C)$ (lemme 10.15). De plus, il existe des cycles orientés $(C', C'') \in V_1^2$ tels que (C', C'') n'est pas un chevauchement dans π (sinon, le lemme 10.12 impliquerait que $K_1(\rho)$ n'a pas d'arête, ce qui contredirait la connexité de $K_1(\rho)$). On définit σ comme étant une inversion agissant sur C'. Le lemme 10.12 implique que σ préserve l'orientation de C'', réduisant ainsi $indice(\sigma)$ d'au moins 1 comparé à $indice(\rho)$, ce qui est une contradiction.

La discussion précédente implique qu'il existe une inversion $\rho \in \mathcal{R}(K)$ vérifiant $indice(\rho) = 0$, i.e. ρ ne crée pas de nouvelle composante non orientée. On obtient donc $\Delta c(\pi, \rho) = 1$ et $\Delta h(\pi, \rho) = 0$, ce qui implique que ρ est solide. ■

10.9 Franchissement des obstacles

Si π a une composante orientée, le théorème 10.9 implique qu'il existe une inversion solide dans π. Dans cette section, on cherche une inversion solide en l'absence de toute composante orientée. Soit $\prec$ un ordre partiel sur un ensemble P. On dit que x est *couvert* par y dans P s'ils vérifient $x \prec y$ et s'il n'existe pas d'élément $z \in P$ avec $x \prec z \prec y$. Le *graphe de couverture* Ω de $\prec$ est un graphe non orienté de sommets P et d'ensemble d'arêtes $\{(x, y) : \ x, y \in P$ et x est couvert par $y\}$.

Soit $\mathcal{U}_\pi$ l'ensemble formé des composantes non orientées de H_π et soit $[U_{min}, U_{max}]$ l'intervalle situé entre les positions extrêmes à gauche et à droite dans une composante non orientée $U \in \mathcal{U}_\pi$. On définit $\overline{U}_{min} = \min_{U \in \mathcal{U}_\pi} U_{min}$ et $\overline{U}_{max} = \max_{U \in \mathcal{U}_\pi} U_{max}$; soit $[\overline{U}_{min}, \overline{U}_{max}]$ l'intervalle situé entre les positions extrême gauche et extrême droite parmi toutes les composantes non orientées de π. Soit $\overline{U}$ une composante (*artificielle*) associée à l'intervalle $[\overline{U}_{min}, \overline{U}_{max}]$.

Par définition, $\overline{\mathcal{U}}_\pi$ est l'ensemble de $|\mathcal{U}_\pi| + 1$ éléments constitué des $|\mathcal{U}_\pi|$ éléments $\{U : U \in \mathcal{U}_\pi\}$, auxquels on ajoute un élément supplémentaire $\overline{U}$. Soit $\prec \equiv \prec_\pi$ l'ordre partiel de *confinement* sur $\overline{\mathcal{U}}_\pi$, défini par la règle suivante : $U \prec W$ si et seulement si $[U_{min}, U_{max}] \subset [W_{min}, W_{max}]$, pour $(U, W) \in \overline{\mathcal{U}}_\pi^2$. S'il existe une *plus grande* composante non orientée U dans π (i.e. $[U_{min}, U_{max}] = [\overline{U}_{min}, \overline{U}_{max}]$), on suppose qu'il existe deux éléments (la composante « réelle » U et la composante « artificielle » $\overline{U}$) correspondant au plus grand intervalle et qu'ils vérifient $U \prec_\pi \overline{U}$. Soit Ω_π l'*arbre* représentant le graphe de couverture de l'ordre partiel $\prec_\pi$ sur $\overline{\mathcal{U}}_\pi$ (figure 10.15a). Tout sommet de Ω_π, $\overline{U}$ excepté, est associé à une composante non orientée dans $\mathcal{U}_\pi$. Dans le cas où π présente le plus grand obstacle, on suppose que la feuille $\overline{U}$ lui est associée (autrement dit, dans ce cas, *deux sommets* correspondent au plus grand obstacle : la feuille $\overline{U}$ et sa voisine, le plus grand obstacle $U \in \mathcal{U}_\pi$). Toute feuille de Ω_π correspondant à un élément minimal dans $\prec_\pi$ est un obstacle. Si $\overline{U}$ est une feuille dans Ω_π, ce n'est pas nécessairement un obstacle (par exemple, $\overline{U}$ est une feuille dans Ω_π, mais ce n'est pas un obstacle pour la permutation π montrée dans la figure 10.11a). Par conséquent, le nombre de feuilles dans Ω_π coïncide avec le

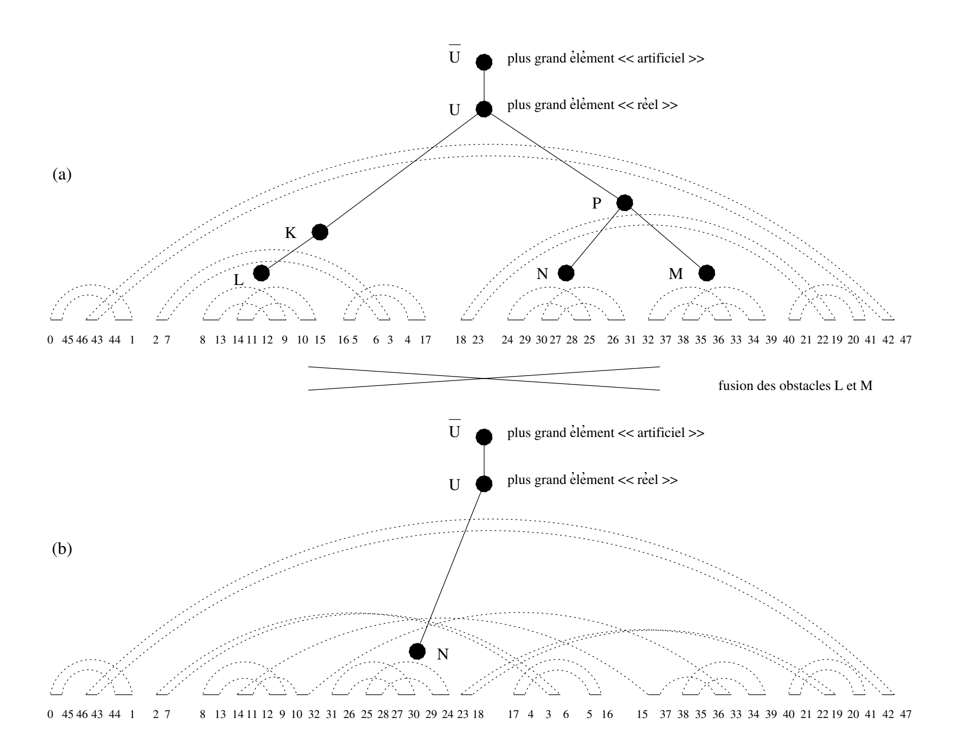


Figure 10.15 – (a) Un graphe de couverture Ω_π d'une permutation π avec les composantes « réelles » non orientées K, L, M, N, P et U et une composante « artificielle » $\overline{U}$; (b) une inversion ρ qui fusionne les obstacles L et M dans π transforme les composantes non orientées L, K, P et M en une composante orientée qui disparaît de $\Omega_{\pi\cdot\rho}$. Cette inversion transforme les cycles non orientés $(32, 33, 36, 37, 32)$ et $(10, 11, 14, 15, 10)$ dans π en un cycle orienté $(15, 14, 11, 10, 32, 33, 36, 37, 15)$ dans $\pi \cdot \rho$. On a $LCA(L, M) = \overline{AMF}(L, M) = U$ et $CHEMIN(A, F) = \{L, K, U, P, M\}$.

nombre d'obstacles $h(\pi)$, sauf[‡]

- lorsqu'il n'existe qu'une composante non orientée dans π (dans ce cas, Ω_π est constitué de deux copies de cette composante et possède deux feuilles, tandis que $h(\pi)$ vaut 1)

- lorsque le plus grand élément est dans $\mathcal{U}_\pi$ et qu'il ne s'agit pas d'un obstacle ; autrement dit, cet élément sépare d'autres obstacles (dans ce cas, le nombre de feuilles vaut $h(\pi) + 1$).

Tout obstacle peut être transformé en une composante orientée par une

[‡]Bien que l'ajout d'une composante « artificielle » $\overline{U}$ puisse sembler un peu étrange et pas indispensable, on va voir par la suite qu'un tel ajout facilite énormément l'analyse des détails techniques.

inversion d'un cycle arbitraire dans cet obstacle (figure 10.10). Une telle opération « coupe » une feuille dans le graphe de couverture, comme le décrit le lemme suivant.

Lemme 10.16 *(Coupe d'obstacles) Toute inversion ρ sur un cycle dans un obstacle K coupe la feuille K du graphe de couverture de π, i.e.* $\Omega_{\pi\cdot\rho} = \Omega_\pi \setminus K$.

Preuve Si ρ agit sur un cycle non orienté d'une composante K dans π, K reste « non cassée » dans $\pi \cdot \rho$. En outre, le lemme 10.9 implique que toute inversion sur un cycle (non orienté) d'une composante (non orientée) K oriente au moins un cycle dans K. Par conséquent, ρ transforme K en une composante orientée dans $\pi \cdot \rho$ et supprime la feuille K du graphe de couverture. ■

Les inversions qui coupent les obstacles ne sont pas toujours solides. Un obstacle $K \in \mathcal{U}_\pi$ *protège* un non-obstacle $U \in \mathcal{U}_\pi$ si la suppression de K de $\mathcal{U}_\pi$ fait passer U du statut de non-obstacle à celui d'obstacle (i.e. U est un obstacle dans $\mathcal{U}_\pi \setminus K$). Un obstacle dans π est un *superobstacle* s'il protège un non-obstacle $U \in \mathcal{U}_\pi$, et un *obstacle simple* dans le cas contraire. Les composantes M, N et U dans la figure 10.15a sont des obstacles simples, alors que la composante L est un superobstacle (supprimer L transforme le non-obstacle K en un obstacle). Dans la figure 10.16a, les trois obstacles sont des superobstacles, tandis que, dans la figure 10.16b, il y a deux superobstacles et un obstacle simple (notons que les graphes de couverture des figures 10.16a et 10.16b sont les mêmes !). Le lemme suivant se déduit immédiatement de la définition d'un obstacle simple.

Lemme 10.17 *Une inversion qui agit sur un cycle d'un obstacle simple est solide.*

Preuve Le lemme 10.16 implique que, pour toute inversion ρ qui agit sur un cycle d'un obstacle simple, on a : $b(\pi) = b(\pi \cdot \rho)$, $c(\pi) = c(\pi \cdot \rho)$ et $h(\pi \cdot \rho) = h(\pi) - 1$. On en déduit que ρ est solide. ■

Malheureusement, une inversion qui agit sur un cycle d'un superobstacle est non solide, car elle transforme un non-obstacle en obstacle, impliquant $\Delta(c - h) = 0$. On va définire une nouvelle opération (fusion d'obstacles) qui permet de rechercher les inversions solides, même en l'absence d'obstacles simples.

Si L et M sont deux obstacles dans π, on définit $CHEMIN(L, M)$ comme étant l'ensemble des composantes (non orientées) sur le chemin (unique) reliant la feuille L à la feuille M dans le graphe de couverture Ω_π. Si L et M sont *tous deux* des éléments *minimaux* pour $\prec$, on définit $AMF(L, M)$ comme étant une composante (non orientée) qui est le *plus petit ancêtre commun* de L et M ; on définit $\overline{AMF}(L, M)$ comme étant le *plus petit ancêtre commun* de L et M *qui ne sépare pas* L et M. Si L correspond au *plus grand* obstacle U, il existe deux éléments U et $\overline{U}$ dans $\overline{\mathcal{U}}_\pi$ qui correspondent au même (plus grand)

intervalle $[U_{min}, U_{max}] = [\overline{U}_{min}, \overline{U}_{max}]$. Dans ce cas, on définit $AMF(L, M) = \overline{AMF}(L, M) = U$. Soient $G(V, E)$ un graphe, $w \in V$ et $W \subset V$. Par définition, une *contraction de W en w dans G* est un nouveau graphe d'ensemble de sommets $V \setminus (W \setminus w)$ et d'ensemble d'arêtes $\{(p(x), p(y)) : (x, y) \in E^2\}$, avec $p(v) = w$ si v appartient à W et $p(v) = v$ sinon. Notons que, si w appartient à W, une contraction réduit le nombre de sommets dans G de $|W| - 1$, tandis que, dans le cas contraire, le nombre de sommets est réduit de $|W|$.

Soient L et M deux obstacles dans π et soit Ω_π le graphe de couverture de π. On définit $\Omega_\pi(L, M)$ comme étant le graphe obtenu à partir de Ω_π par la contraction de $CHEMIN(L, M)$ en $\overline{AMF}(L, M)$ (on ignore les boucles dans le graphe $\Omega_\pi(L, M)$). Notons que, dans le cas

$$AMF(L, M) = \overline{AMF}(L, M),$$

$\Omega_\pi(L, M)$ correspond à la suppression des éléments de $CHEMIN(L, M) \setminus ACM(L, M)$ de l'ordre partiel $\prec_\pi$, tandis que lorsque l'on a

$$AMF(L, M) \neq \overline{AMF}(L, M),$$

$\Omega_\pi(L, M)$ correspond à la suppression de l'ensemble complet $CHEMIN(L, M)$ de $\prec_\pi$.

Lemme 10.18 *(Fusion d'obstacles) Soit π une permutation de graphe de couverture Ω_π et soit ρ une inversion qui agit sur les arêtes noires de (différents) obstacles L et M dans π. Alors ρ agit sur Ω_π comme la contraction de $CHEMIN(L, M)$ en $\overline{AMF}(L, M)$, i.e. $\Omega_{\pi \cdot \rho} = \Omega_\pi(L, M)$.*

Preuve L'inversion ρ agit sur les arêtes noires des cycles $C_1 \in L$ et $C_2 \in M$ dans $G(\pi)$ et transforme C_1 et C_2 en un cycle orienté C dans $G(\pi \cdot \rho)$ (figure 10.15). Il est facile de vérifier que tout cycle qui chevauche C_1 ou C_2 dans $G(\pi)$ chevauche C dans $G(\pi \cdot \rho)$. Ceci implique que ρ transforme les obstacles L et M dans π en différentes parties d'une composante orientée dans $\pi \cdot \rho$; par conséquent, L et M « disparaissent » de $\Omega_{\pi \cdot \rho}$.

En outre, toute composante de $CHEMIN(L, M) \setminus \overline{AMF}(L, M)$ possède au moins un cycle qui chevauche C dans $G(\pi \cdot \rho)$. Ceci implique que toute composante de ce type dans π devient une partie d'une composante orientée dans $\pi \cdot \rho$; elle « disparaît » donc de $\Omega_{\pi \cdot \rho}$. Toute composante de $\mathcal{U}_\pi \setminus CHEMIN(L, M)$ demeure non orientée dans $\pi \cdot \rho$. La composante $AMF(L, M)$ reste non orientée si et seulement si elle vérifie $AMF(L, M) = \overline{AMF}(L, M)$. Toute composante qui est recouverte par un sommet de $CHEMIN(L, M)$ dans $\prec_\pi$ sera recouverte par $\overline{AMF}(L, M)$ dans $\prec_{\pi \cdot \rho}$. ■

On écrit $U < W$ pour les obstacles U et W si la position extrême droite de U est plus petite que la position extrême droite de W, i.e. $U_{max} < W_{max}$. On range les obstacles de π dans l'ordre croissant de leur position extrême droite

$$U(1) < \ldots < U(l) \equiv L < \ldots < U(m) \equiv M < \ldots < U(h(\pi))$$

et on définit les ensembles d'obstacles

$$ENTRE(L,M) = \{U(i) :\ l < i < m\}$$

et

$$EXT(L,M) = \{U(i) :\ i \notin [l,m]\}.$$

Lemme 10.19 *On note ρ une inversion qui fusionne les obstacles L et M dans π. Si les deux ensembles d'obstacles $ENTRE(L,M)$ et $EXT(L,M)$ sont non vides, alors ρ est solide.*

Preuve Soient $U' \in ENTRE(L,M)$ et $U'' \in EXT(L,M)$. Le lemme 10.18 implique que l'inversion ρ supprime les obstacles L et M de Ω_π. Elle risque aussi d'ajouter un nouvel obstacle K dans $\pi \cdot \rho$ en faisant passer K de non-obstacle dans π en obstacle dans $\pi \cdot \rho$. Si tel est le cas, K ne sépare pas L et M dans π (sinon, d'après le lemme 10.18, K serait supprimé de $\pi \cdot \rho$). Sans perte de généralité, on suppose $L < U' < M$.

Si K est un obstacle *minimal* dans $\pi \cdot \rho$, on a $L \prec_\pi K$ *ou* $M \prec_\pi K$ (sinon, K serait un obstacle dans π). Comme K ne sépare pas L et M dans π, on a $L \prec_\pi K$ *et* $M \prec_\pi K$. Comme U' est pris en sandwich entre L et M, on en déduit $U' \prec_\pi K$, d'où $U' \prec_{\pi\cdot\rho} K$, ce qui contredit la minimalité de K dans $\pi \cdot \rho$.

Si K est le *plus grand* obstacle dans $\pi \cdot \rho$, on a $L, M \not\prec_\pi K$ ou $L, M \prec_\pi K$ (si l'on était dans le cas $L \not\prec_\pi K$ et $M \prec_\pi K$, alors, d'après le lemme 10.18, K serait supprimé de $\pi \cdot \rho$). Si l'on a $L, M \not\prec_\pi K$, on obtient $L < U' \prec_\pi K < M$, i.e. K est coincé entre L et M. Par conséquent, U'' se trouve à l'extérieur de K dans π et vérifie $U'' \not\prec_{\pi\cdot\rho} K$, une contradiction. Si l'on a $L, M \prec_\pi K$, alors, comme K est un non-obstacle dans π, K sépare L, M d'un autre obstacle N. Donc K sépare U' de N. Comme N et U' « survivent » dans $\pi \cdot \rho$, K sépare N et U' dans $\pi \cdot \rho$, ce qui est contradictoire.

Par conséquent, ρ supprime les obstacles L et M de Ω_π et n'ajoute pas de nouvel obstacle dans $\pi \cdot \rho$, d'où $\Delta h = -2$. Comme $b(\pi \cdot \rho) = b(\pi)$ et $c(\pi \cdot \rho) = c(\pi) - 1$, on obtient $\Delta(b - c + h) = -1$ et l'inversion ρ est solide. ■

Lemme 10.20 *Si $h(\pi)$ est strictement supérieur à 3, il existe une inversion solide qui fusionne deux obstacles dans π.*

Preuve On range $h(\pi)$ obstacles de π selon l'ordre croissant de leur position extrême droite ; soient L et M le premier et le $(1 + \frac{h(\pi)}{2})$-ième obstacles, dans cet ordre. Comme on sait que $h(\pi) > 3$, les deux ensembles $ENTRE(L,M)$ et $EXT(L,M)$ sont non vides et, d'après le lemme 10.19, l'inversion ρ qui fusionne L et M est solide. ■

Lemme 10.21 *Si $h(\pi) = 2$, il existe une inversion solide qui fusionne deux obstacles dans π. Si $h(\pi) = 1$, il existe une inversion solide qui coupe le seul obstacle dans π.*

Preuve Si $h(\pi) = 2$, soit Ω_π est un graphe de chemin, soit il contient la plus grande composante séparant deux obstacles dans π. Dans les deux cas, la fusion des obstacles dans π est une inversion solide (lemme 10.18). Si $h(\pi) = 1$, le lemme 10.16 fournit une inversion solide qui coupe le seul obstacle dans π. ■

Les lemmes précédents montrent que les obstacles qui fusionnent fournissent une façon de trouver des inversions solides, même en l'absence d'obstacles simples. En revanche, les obstacles fusionnants ne fournissent pas un moyen de transformer un superobstacle en un obstacle simple.

Lemme 10.22 *Soit ρ une inversion dans π qui fusionne deux obstacles L et M. Alors tout superobstacle dans π (différent de L et M) reste un superobstacle dans $\pi \cdot \rho$.*

Preuve Soit U un superobstacle dans π (différent de L et M) qui protège un non-obstacle U'. Il est clair que, si U' est un obstacle minimal dans $\mathcal{U}_\pi \setminus U$, U reste un superobstacle dans $\pi \cdot \rho$. Si U' est le plus grand obstacle dans $\mathcal{U}_\pi \setminus U$, alors U' ne sépare aucun obstacle dans $\mathcal{U}_\pi \setminus U$. Par conséquent, U' n'appartient pas à $CHEMIN(L, M)$; il « survit » donc dans $\pi \cdot \rho$ (lemme 10.18). Ceci implique que U' reste protégé par U dans $\pi \cdot \rho$. ■

10.10 Théorème de dualité pour la distance d'inversion

À moins que Ω_π ne soit homéomorphe à l'*étoile d'ordre 3* (le graphe avec trois arêtes incidentes au même sommet), les lemmes 10.20 et 10.21 impliquent qu'il existe une inversion solide dans π. Par ailleurs, si au moins un obstacle dans π est simple, alors le lemme 10.17 implique qu'il existe une inversion solide dans π. Ainsi, le seul cas dans lequel il peut ne pas exister d'inversion solide est celui où Ω_π est homéomorphe à l'étoile d'ordre 3 avec trois superobstacles ; on l'appelle une *3-forteresse* (figure 10.16b).

Lemme 10.23 *Si ρ est une inversion qui détruit une 3-forteresse π (autrement dit, $\pi \cdot \rho$ n'est pas une 3-forteresse), alors ρ n'est pas solide.*

Preuve Toute inversion sur une permutation π peut réduire $h(\pi)$ d'au plus 2 et la *seule* opération pouvant réduire le nombre d'obstacles de 2 est la fusion d'obstacles. D'un autre côté, le lemme 10.18 implique que la fusion d'obstacles dans une 3-forteresse peut réduire $h(\pi)$ d'au plus 1. Par conséquent : $\Delta h \geq -1$.

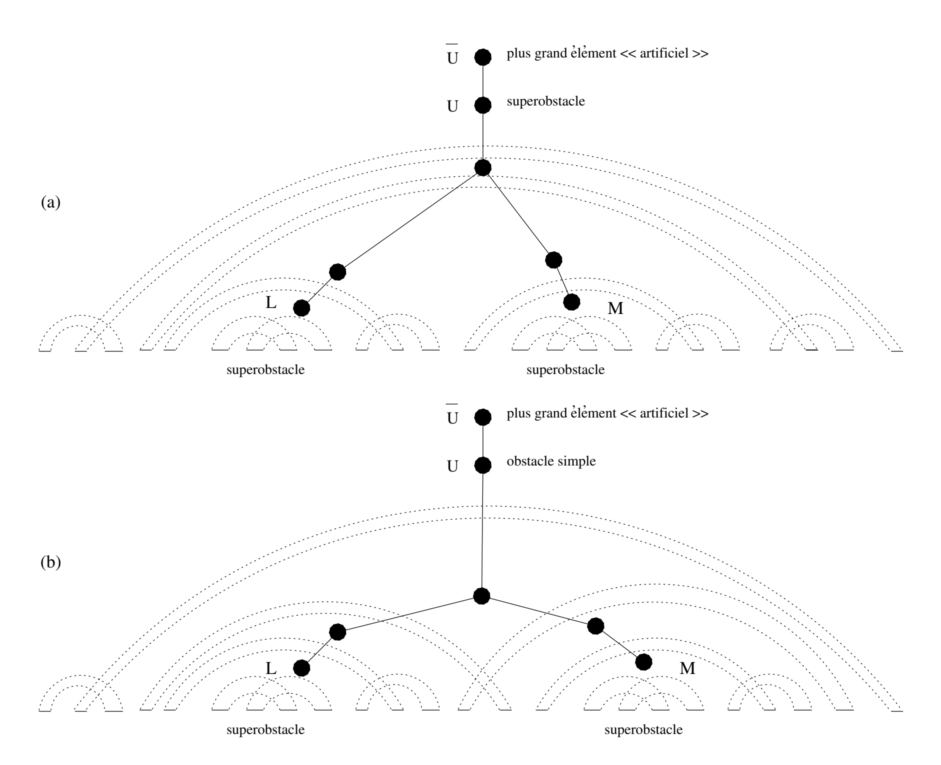


Figure 10.16 – La permutation (a) est une 3-forteresse, tandis que la permutation (b), pourtant associée au même graphe de couverture, n'en est pas une (l'obstacle U n'est pas un superobstacle).

Notons que, pour toute inversion n'agissant pas sur les arêtes du *même* cycle : $\Delta c = -1$; on en déduit que toute inversion qui n'agit pas sur les arêtes du même cycle dans une 3-forteresse est non solide.

Si ρ agit sur un cycle dans une composante non orientée d'une 3-forteresse, alors elle ne réduit pas le nombre d'obstacles. Comme Δc est nul pour une inversion sur un cycle non orienté, ρ est non solide.

Si ρ agit sur un cycle dans une composante orientée d'une 3-forteresse, alors elle ne détruit aucune composante non orientée dans π et elle ne réduit pas le nombre d'obstacles. Si ρ augmente le nombre d'obstacles, alors $\Delta h \geq 1$ et $\Delta c \leq 1$ impliquent que ρ n'est pas solide. Si le nombre d'obstacles dans $\pi \cdot \rho$ reste le même, alors tout superobstacle dans π demeure un superobstacle dans $\pi \cdot \rho$, impliquant ainsi que $\pi \cdot \rho$ est une 3-forteresse, ce qui est contradictoire. ■

Lemme 10.24 *Si π est une 3-forteresse, elle vérifie l'égalité $d(\pi) = n + 1 - c(\pi) + h(\pi) + 1$.*

Preuve Le lemme 10.23 implique que tout tri d'une 3-forteresse comporte au moins une inversion non solide. D'où $d(\pi) \geq b(\pi) - c(\pi) + h(\pi) + 1$.

Si π possède des cycles orientés, toutes ses composantes orientées peuvent être détruites par des remplissages solides (théorème 10.7) et des inversions solides dans les composantes orientées (théorème 10.9), sans pour autant affecter les composantes non orientées.

Si π est une 3-forteresse sans cycle orienté, alors une inversion (non solide) ρ qui fusionne des obstacles arbitraires dans π aboutit à une permutation $\pi \cdot \rho$ avec deux obstacles (lemme 10.18). Une fois encore, les cycles orientés qui apparaissent dans $\pi \cdot \rho$ après une telle fusion peuvent être détruits par des remplissages solides et des inversions solides dans les composantes orientées (théorèmes 10.7 et 10.9), aboutissant à une permutation σ avec $h(\sigma) = 2$. Les théorèmes 10.7 et 10.9 et le lemme 10.21 impliquent que σ peut être triée par des remplissages solides et des inversions solides. Il existe donc un tri généralisé de π dans lequel tous les remplissages et toutes les inversions sauf une sont solides. Par conséquent, ce tri généralisé contient $n+1-c(\pi)+h(\pi)+1$ inversions. Le lemme 10.11 implique que le tri généralisé de π simule un (véritable) tri optimal de π par $d(\pi) = n+1-c(\pi)+h(\pi)+1$ inversions. ■

Dans la suite, on tente d'éviter la création de 3-forteresses au cours du tri par inversions. Si l'on y parvient, la permutation π peut être triée en $n+1-c(\pi)+h(\pi)$ inversions. Sinon, on montre comment trier π en $n+1-c(\pi)+h(\pi)+1$ inversions et on prouve que de telles permutations ne peuvent être triées avec moins d'inversions. Une permutation π est appelée une *forteresse* si elle possède un nombre impair d'obstacles et si tous ces obstacles sont des superobstacles.

Lemme 10.25 *Si ρ est une inversion qui détruit une forteresse π avec $h(\pi)$ superobstacles (i.e. $\pi \cdot \rho$ n'est pas une forteresse avec $h(\pi)$ superobstacles), alors soit ρ est non solide, soit $\pi \cdot \rho$ est une forteresse avec $h(\pi) - 2$ superobstacles.*

Preuve Toute inversion agissant sur une permutation peut réduire le nombre d'obstacles d'au plus deux ; la *seule* opération capable de réduire le nombre d'obstacles de deux est une fusion d'obstacles. Des arguments similaires à ceux de la preuve du lemme 10.23 démontrent que, si ρ ne fusionne pas d'obstacles, alors ρ n'est pas solide. Si une inversion solide ρ fusionne des (super)obstacles L et M dans π, alors le lemme 10.18 implique que cette inversion réduit le nombre d'obstacles de deux et, si $h(\pi)$ est strictement supérieur à 3, elle ne crée pas de nouvel obstacle. Le lemme 10.22 implique que tout superobstacle dans π (sauf L et M) reste un superobstacle dans $\pi \cdot \rho$, ce qui implique que $\pi \cdot \rho$ est une forteresse avec $h(\pi) - 2$ superobstacles. ■

Lemme 10.26 *Si π est une forteresse, alors elle vérifie l'inégalité $d(\pi) \geq n + 1 - c(\pi) + h(\pi) + 1$.*

Preuve Le lemme 10.25 implique que tout tri de π contient une inversion non solide ou qu'il diminue graduellement le nombre de superobstacles dans π en transformant une forteresse avec h (super)obstacles en une forteresse avec

$h-2$ (super)obstacles. Par conséquent, si un tri de π n'utilise que des inversions solides, il aboutira éventuellement à une 3-forteresse. D'après le lemme 10.23, tout tri d'une forteresse contient donc au moins une inversion non solide, d'où $d(\pi) \geq n + 1 - c(\pi) + h(\pi) + 1$. ■

Formulons enfin le théorème de dualité pour le tri par inversions de permutations signées :

Théorème 10.10 *(Hannenhalli et Pevzner, 1995 [154]) Pour toute permutation π, on a :*

$$d(\pi) = \begin{cases} n + 1 - c(\pi) + h(\pi) + 1, & \textit{si } \pi \textit{ est une forteresse} \\ n + 1 - c(\pi) + h(\pi), & \textit{sinon.} \end{cases}$$

Preuve Si π possède un nombre pair d'obstacles, les remplissages solides (théorème 10.7), les inversions solides dans des composantes orientées (théorème 10.9) et les obstacles solides fusionnants (lemmes 10.20 et 10.21) mènent à un tri généralisé de π par $n + 1 - c(\pi) + h(\pi)$ inversions.

Si π possède un nombre impair d'obstacles, au moins l'un d'eux est simple ; il existe donc une inversion solide qui coupe cet obstacle simple (lemme 10.17). Cette inversion simple aboutit à une permutation avec un nombre pair d'obstacles. Par conséquent, de façon similaire au cas précédent, il existe un tri généralisé de π qui n'utilise que des remplissages solides et $n + 1 - c(\pi) + h(\pi)$ inversions solides.

Par conséquent, si π n'est pas une forteresse, il existe un tri généralisé de π par $n+1-c(\pi)+h(\pi)$ inversions. Le lemme 10.11 implique que ce tri généralisé simule un (véritable) tri optimal de π.

Si π est une forteresse, il existe une suite de remplissages solides (théorème 10.7), d'inversions solides dans des composantes orientées (théorème 10.9) et d'obstacles solides fusionnants (lemme 10.20) qui mène à une 3-forteresse pouvant être triée par une série d'inversions avec au plus une inversion non solide. On en déduit qu'il existe un tri généralisé de π qui utilise $n+1-c(\pi)+h(\pi)+1$ inversions. Le lemme 10.26 implique que ce tri généralisé simule le (véritable) tri optimal de π avec $d(\pi) = n + 1 - c(\pi) + h(\pi) + 1$ inversions. ■

Ce théorème explique le mystère de l'étonnante performance des algorithmes d'approximation pour le tri par inversions de permutations signées. Une explication simple de cette performance est que la borne $d(\pi) \geq n + 1 - c(\pi)$ est extrêmement bien ajustée, car $h(\pi)$ est petit pour des permutations choisies au hasard.

10.11 Algorithme de tri par inversions

Les lemmes 10.11, 10.17, 10.20 et 10.21, ainsi que les théorèmes 10.7, 10.9 et 10.10 motivent l'algorithme *Tri_Inversions*, qui trie de façon optimale les permutations signées.

Tri_Inversions(π)
tant que π n'est pas trié
  si π a un cycle long
    choisir un (g, b)-remplissage solide ρ de π (théorème 10.7)
  sinon si π a une composante orientée
    choisir une inversion solide ρ dans cette composante (théorème 10.9)
  sinon si π a un nombre pair d'obstacles
    choisir une inversion solide ρ qui fusionne deux obstacles dans π (lemmes 10.20 et 10.21)
  sinon si π a au moins un obstacle simple
    choisir une inversion solide ρ qui coupe cet obstacle dans π (lemmes 10.17 et 10.21)
  sinon si π est une forteresse avec plus de trois superobstacles
    choisir une inversion solide ρ qui fusionne deux (super)obstacles dans π (lemme 10.20)
  sinon /* π est une 3-forteresse */
    choisir une inversion (non) solide ρ qui fusionne deux (super)obstacles arbitraires dans π
  $\pi \leftarrow \pi \cdot \rho$
fin du « tant que »
simuler un (véritable) tri de π par le tri généralisé calculé de π (lemme 10.11)

Théorème 10.11 *L'algorithme* $Tri_Inversions(\pi)$ *trie de façon optimale une permutation* π *d'ordre* n *en un temps* $O(n^4)$.

Preuve Le théorème 10.10 implique que *Tri_Inversions* fournit un tri généralisé de π par une série d'inversions et de remplissages contenant $d(\pi)$ inversions. D'après le lemme 10.11, ce tri généralisé simule un (véritable) tri optimal de π par $d(\pi)$ inversions.

Notons que toute itération de la boucle `tant que` dans *Tri_Inversions* réduit la quantité $complexit\acute{e}(\pi) + 3d(\pi)$ d'au moins 1, ce qui implique que le nombre d'itérations de *Tri_Inversions* est borné par $4n$. L'itération la plus « coûteuse » est la recherche d'une inversion solide dans une composante orientée. Comme cet algorithme peut être implémenté en un temps $O(n^3)$ pour les permutations simples, la complexité temporelle globale de *Tri_Inversions* est $O(n^4)$. ■

Ci-dessous, on décrit une version simplifiée de *Tri_Inversions*, qui n'utilise pas les remplissages et fonctionne en un temps $O(n^5)$. On définit :

$$f(\pi) = \begin{cases} 1, & \text{si } \pi \text{ est une forteresse} \\ 0, & \text{sinon.} \end{cases}$$

Une inversion ρ est dite *valide* si elle vérifie $\Delta(c-h-f)=1$. Les preuves du théorème 10.6 et du lemme 10.26 impliquent l'inégalité $\Delta(c-h-f)\geq -1$. Cette observation et le théorème 10.10 mènent au résultat suivant :

Théorème 10.12 *Dans toute permutation π, il existe une inversion valide. Toute suite d'inversions valides triant π est optimale.*

Le théorème 10.12 motive la version simple de $Tri_Inversions$, qui est très rapide dans la pratique :

Tri_Inversions_Simple(π)
1. **tant que** π n'est pas trié
2. choisir une inversion valide ρ dans π (théorème 10.12)
3. $\pi \leftarrow \pi \cdot \rho$
4. **fin du « tant que »**

10.12 Transformation d'hommes en souris

L'analyse des réarrangements dans les génomes multichromosomiques utilise le théorème de dualité pour les génomes monochromosomiques (Hannenhalli et Pevzner, 1995 [154]) et deux notions supplémentaires appelées *retournement* et *coiffage* de chromosomes. Dans les études portant sur les réarrangements dans les génomes multichromosomiques, un *chromosome* est défini comme une *séquence* de gènes et un *génome* comme un *ensemble* de chromosomes. Soit $\Pi=\{\pi(1),\ldots,\pi(N)\}$ un génome constitué de N chromosomes et soit $\pi(i)=\pi(i)_1\ldots\pi(i)_{n_i}$, où n_i est le nombre de gènes dans le i-ième chromosome. Tout chromosome π peut être lu soit « de gauche à droite » (c'est-à-dire comme $\pi=\pi_1\ldots\pi_n$), soit « de droite à gauche » (c'est-à-dire comme $-\pi=-\pi_n\ldots-\pi_1$), ce qui mène à deux représentations équivalentes du même chromosome. Vu sous cet angle, un génome de trois chromosomes $\{\pi(1),\pi(2),\pi(3)\}$ est équivalent à $\{\pi(1),-\pi(2),\pi(3)\}$ ou à $\{-\pi(1),\pi(2),-\pi(3)\}$, autrement dit les *directions* des chromosomes n'ont pas d'importance. Dans les génomes multichromosomiques, les quatre événements élémentaires de réarrangement les plus fréquents sont les *inversions*, les *translocations*, les *fusions* et les *fissions*.

On distingue les inversions *internes*, qui n'impliquent pas les extrémités des chromosomes (c'est-à-dire les inversions $\rho(\pi,i,j)$ d'un chromosome π à n gènes avec $1<i\leq j<n$) et les inversions *préfixes*, qui impliquent les extrémités des chromosomes (c'est-à-dire $i=1$ ou $j=n$). Une translocation est dite *interne* si ce n'est ni une fusion ni une fission.

Pour un chromosome $\pi=\pi_1\ldots\pi_n$, les nombres $+\pi_1$ et $-\pi_n$ sont appelés les *queues* de π. Notons que le changement d'orientation d'un chromosome ne change pas l'ensemble de ses queues. Les queues d'un génome Π à N chromosomes contiennent l'ensemble $\mathcal{T}(\Pi)$ de $2N$ queues. Dans cette section, on considère des génomes Π et Γ *à queue commune*, avec $\mathcal{T}(\Pi)=\mathcal{T}(\Gamma)$.

Pour des génomes à queue commune, les inversions internes et les translocations suffisent pour le tri génomique — autrement dit, les inversions de préfixes, les fusions et les fissions peuvent être ignorées (la véracité de cette assertion deviendra claire par la suite). Pour des chromosomes $\pi = \pi_1 \ldots \pi_n$ et $\sigma = \sigma_1 \ldots \sigma_m$, on note $\pi + \sigma$ la fusion $\pi_1 \ldots \pi_n \sigma_1 \ldots \sigma_m$ et $\pi - \sigma$ la fusion $(\pi_1 \ldots \pi_n - \sigma_m \ldots - \sigma_1)$. Étant donnés un ordonnancement des chromosomes $(\pi(1), \ldots, \pi(N))$ dans un génome Π et un vecteur de *retournement* $s = (s(1), \ldots, s(N))$ avec $s(i) \in \{-1, +1\}$, on peut former un *concaténé* de Π sous la forme d'une permutation $\Pi(s) = s(1)\pi(1) + \ldots + s(N)\pi(N)$ sur $\sum_{i=1}^{N} n_i$ éléments. En fonction du choix d'un vecteur de retournement, il existe 2^N concaténés de Π pour chacun des $N!$ ordonnancements des chromosomes dans Π. Si l'ordre des chromosomes dans un génome Π est fixé, on dit que Π est un génome *ordonné*.

Dans cette section, on suppose, sans perte de généralité, que $\Gamma = (\gamma_1, \ldots, \gamma_N)$ est un génome (ordonné) et on note $\gamma = \gamma_1 + \ldots + \gamma_N$ la permutation identité (autrement dit, on numérote les gènes dans cet ordre). On note $d(\Pi) \equiv d(\Pi, \Gamma)$ et on appelle simplement le problème d'un tri génomique de Π en Γ un *tri d'un génome* Π.

On utilise l'idée suivante pour analyser les génomes à queue commune. Étant donné un concaténé π d'un génome Π, on peut trier π par inversions de façon optimale. Toute inversion dans ce tri correspond à une inversion ou à une translocation dans un tri (pas nécessairement optimal) du génome Π. Par exemple, une translocation $\rho(\pi, \sigma, i, j)$ agissant sur les chromosomes $\pi = \pi_1 \ldots \pi_n$ et $\sigma = \sigma_1 \ldots \sigma_m$ (figure 10.17) peut aussi être vue comme une inversion $\rho(\pi - \sigma, i, n + (m - j + 1))$ qui agit sur $\pi - \sigma$ (et vice versa). Par définition, un *concaténé optimal* de Π est un concaténé π de distance d'inversion $d(\pi)$ minimale parmi tous les concaténés de Π. Ci-dessous, on prouve que le tri d'un concaténé optimal de Π imite un tri optimal d'un génome Π. Cette approche réduit le problème du tri de Π à celui qui consiste à trouver un concaténé optimal de Π.

Dans la suite, lorsque nous évoquerons le nombre de cycles, il s'agira du nombre de cycles de longueur supérieure à 2, i.e. $c(\pi) - a(\pi)$, où $a(\pi)$ est le nombre d'adjacences (toute adjacence est un cycle de longueur 2). Comme on a $b(\pi) = n + 1 - a(\pi)$, le théorème de Hannenhalli-Pevzner peut être reformulé sous la forme $d(\pi) = b(\pi) - c(\pi) + h(\pi) + f(\pi)$, où $f(\pi)$ vaut 1 si π est une forteresse et 0 dans le cas contraire.

Soit π un concaténé de $\Pi = (\pi(1), \ldots, \pi(N))$. Toute queue de $\pi(i)$ correspond à deux sommets du graphe de points de rupture $G(\pi)$, dont un exactement est un sommet du bord (soit à l'extrême gauche, soit à l'extrême droite) parmi les sommets du chromosome $\pi(i)$ dans le concaténé π. On utilise le mot *queue* pour désigner de tels sommets dans $G(\pi)$. Une arête dans un graphe de points de rupture $G(\pi)$ d'un concaténé π est dite *interchromosomique* si elle relie des sommets situés dans différents chromosomes de Π ; sinon, on dit qu'elle est *intrachromosomique*. Une composante de π est dite *interchromosomique* si elle contient une arête interchromosomique et *intrachromosomique* sinon.

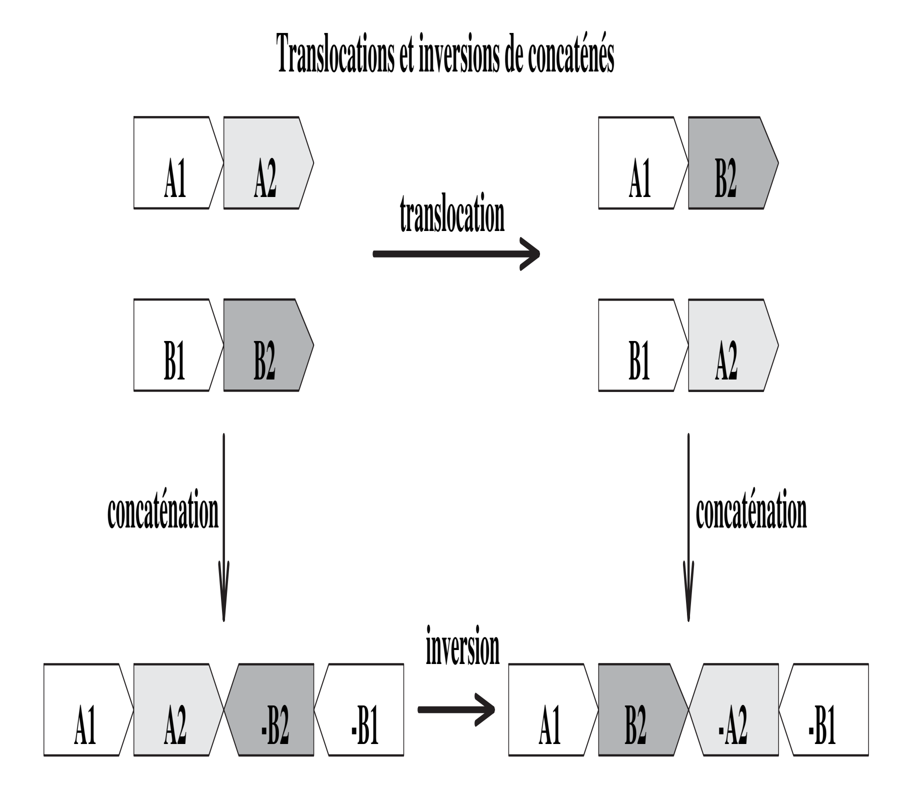


Figure 10.17 – Les translocations peuvent être imitées par les inversions dans un génome concaténé.

Toute arête noire interchromosomique dans $G(\pi)$ relie deux queues. Soit $b_{queue}(\Pi)$ le nombre d'arêtes noires interchromosomiques dans $G(\pi)$ (on sait : $b_{queue}(\Pi) = N - 1$). Notons que, pour des génomes à queue commune, les queues dans $G(\Pi)$ ne sont adjacentes qu'à des queues ; un cycle qui comporte une queue ne contient donc que des queues. Soit $c_{queue}(\Pi)$ le nombre de cycles de $G(\pi)$ qui contiennent des queues. On définit $b(\Pi) = b(\pi) - b_{queue}(\pi)$ (notons les égalités $b(\Pi) = n - N$ et $b(\pi) = n - 1$) et $c(\Pi) = c(\pi) - c_{queue}(\pi)$.

Considérons l'ensemble des composantes intrachromosomiques non orientées $\mathcal{IU}_\pi$ dans π. Les obstacles, les superobstacles et les forteresses pour l'ensemble $\mathcal{IU}_\pi$ sont appelés respectivement des *nœuds*, des *supernœuds* et des *forteresses de nœuds*. Soit $k(\Pi)$ le nombre de nœuds dans un concaténé π de Π. On définit $f(\Pi) = 1$ si π est une forteresse de nœuds et $f(\Pi) = 0$ sinon. Il est clair que $b(\Pi)$, $c(\Pi)$, $k(\Pi)$ et $f(\Pi)$ ne dépendent pas du choix d'un concaténé π.

Lemme 10.27 *Pour des génomes à queue commune* Π *et* Γ*, on a :*

$$d(\Pi) \geq b(\Pi) - c(\Pi) + k(\Pi) + f(\Pi)$$

.

Preuve La preuve est une version plus évoluée de la preuve du lemme 10.26. ■

Des concaténés $\Pi(s)$ et $\Pi(s')$ d'un génome (ordonné) Π sont des *i-jumeaux* si les orientations de tous les chromosomes à l'exception du i-ième dans $\Pi(s)$ et $\Pi(s')$ coïncident ; autrement dit, $s(i) = -s'(i)$ et $s(j) = s'(j)$ pour $j \neq i$. On dit qu'un chromosome $\pi(i)$ est *proprement retourné* dans $\Pi(s)$ si toutes les arêtes interchromosomiques provenant de ce chromosome appartiennent à des composantes orientées dans $\Pi(s)$. Un concaténé π est *proprement retourné* si chaque chromosome dans π est proprement retourné. Le lemme suivant prouve l'existence d'un concaténé proprement retourné.

Lemme 10.28 *Si un chromosome $\pi(i)$ n'est pas proprement retourné dans $\pi = \Pi(s)$, alors il est proprement retourné dans le i-jumeau π' de π. En outre, tout chromosome proprement retourné dans π reste proprement retourné dans π'.*

Preuve Soit g une arête grise interchromosomique dans π provenant du chromosome $\pi(i)$ qui appartient à une composante non orientée dans π. Notons que l'orientation de toute arête grise interchromosomique provenant de $\pi(i)$ est différente dans π et dans π' (i.e. une arête non orientée dans π devient orientée dans π', et vice versa). Comme toutes les arêtes qui chevauchent g dans π sont non orientées, toute arête interchromosomique qui provient de $\pi(i)$ et qui chevauche g dans π est orientée dans π'.

Toutes les arêtes interchromosomiques provenant de $\pi(i)$ qui ne chevauchent pas g dans π chevauchent g dans π'. Comme g est orientée dans π', toutes ces arêtes appartiennent à une composante orientée contenant g dans π'. Par conséquent, $\pi(i)$ est proprement retourné dans π'.

Soit $\pi(j)$ un chromosome proprement retourné dans π. Si $\pi(j)$ n'est pas proprement retourné dans π', il existe une composante interchromosomique non orientée U avec une arête grise interchromosomique provenant de $\pi(j)$ dans π'. Si U n'a pas d'arête issue de $\pi(i)$ dans π', alors U est une composante non orientée dans π, ce qui implique que $\pi(j)$ n'était pas proprement retourné dans π, une contradiction. Si U possède une arête grise (non orientée) h provenant de $\pi(i)$, il est clair que cette arête ne chevauche pas g dans π'. Par conséquent, h chevauche g dans π et h est orientée dans π, ce qui implique que g appartenait à une composante orientée dans π et on aboutit de nouveau à une contradiction. ■

Le lemme 10.28 implique l'existence d'un concaténé proprement retourné $\pi = \Pi(s)$ avec $h(\pi) = k(\Pi)$ et $f(\pi) = f(\Pi)$. On montre maintenant qu'il existe un tri de π par $b(\pi) - c(\pi) + h(\pi) + f(\pi)$ inversions qui simule le tri de Π par $b(\Pi) - c(\Pi) + k(\Pi) + f(\Pi)$ inversions internes et translocations.

Théorème 10.13 *(Hannenhalli et Pevzner, 1995 [153]) Pour des génomes à queue commune Π et Γ, on a : $d(\Pi, \Gamma) \equiv d(\Pi) = b(\Pi) - c(\Pi) + k(\Pi) + f(\Pi)$.*

Preuve Raisonnons par l'absurde et supposons le contraire ; soit Π un génome ayant une valeur minimale égale à $b(\Pi)-c(\Pi)+h(\Pi)+f(\Pi)$ parmi les génomes pour lesquels le théorème est faux. Soit π un concaténé proprement retourné de Π avec une valeur minimale égale à $b_{queue}(\pi) - c_{queue}(\pi)$ parmi tous les concaténés proprement retournés de Π.

Si $b_{queue}(\pi)$ est égal à $c_{queue}(\pi)$ (i.e. toute arête noire interchromosomique est impliquée dans un cycle de longueur 2), il existe un tri optimal de π par $b(\pi) - c(\pi) + k(\pi) + f(\pi)$ inversions qui agissent sur les arêtes noires intrachromosomiques (Hannenhalli et Pevzner, 1995 [154]). Toute inversion ρ de ce type peut être simulée comme une inversion interne ou une translocation interne sur Π, aboutissant ainsi à un tri de Π par $b(\pi) - c(\pi) + k(\pi) + f(\pi)$ inversions/translocations internes. Comme π est un concaténé proprement retourné, il vérifie $b(\pi) = b(\Pi) + b_{queue}(\pi)$, $c(\pi) = c(\Pi) + c_{queue}(\pi)$, $h(\pi) = k(\Pi)$ et $f(\pi) = f(\Pi)$. Par conséquent, le tri optimal de π imite un tri optimal de Π par $b(\Pi) - c(\Pi) + k(\Pi) + f(\Pi)$ inversions/translocations internes.

Si $b_{queue}(\pi)$ est strictement supérieur à $c_{queue}(\pi)$, il existe une arête noire interchromosomique impliquée dans un cycle de longueur supérieure à 2 et elle appartient à une composante orientée dans π (car toute arête noire interchromosomique appartient à une composante orientée dans un concaténé proprement retourné). Hannenhalli et Pevzner, 1995 [154] ont prouvé que, s'il existe une composante orientée dans π, il existe une inversion ρ dans π qui agit sur les arêtes noires d'un cycle orienté de cette composante et qui vérifie $c(\pi \cdot \rho) = c(\pi) + 1$. De plus, cette inversion ne crée pas de nouvelle composante non orientée dans $\pi \cdot \rho$; elle vérifie $h(\pi \cdot \rho) = h(\pi)$ et $f(\pi \cdot \rho) = f(\pi)$. Notons que tout cycle qui contient des queues de chromosomes appartient à une composante orientée dans π et est entièrement constitué d'arêtes situées entre des queues. Par conséquent, ρ agit soit sur deux arêtes noires intrachromosomiques, soit sur deux arêtes noires interchromosomiques appartenant à un cycle orienté de cette composante.

Une inversion ρ agissant sur deux arêtes noires interchromosomiques peut être vue comme une transformation d'un concaténé π de

$$\Pi = (\pi(1), \ldots, \pi(i-1), \pi(i), \ldots, \pi(j), \pi(j+1), \ldots, \pi(N))$$

en un concaténé $\pi \cdot \rho' = \Pi'(s')$, où Π' est le nouvel ordonnancement :

$$(\pi(1), \ldots, \pi(i-1), \pi(j), \ldots, \pi(i), \pi(j+1), \ldots, \pi(N))$$

des chromosomes et où s' est :

$$(s(1), \ldots, s(i-1), -s(j), \ldots, -s(i), s(j+1), \ldots, s(N)).$$

Par conséquent, on obtient $b_{queue}(\pi \cdot \rho) - c_{queue}(\pi \cdot \rho) = b_{queue}(\pi) - (c_{queue}(\pi) + 1)$ et $\pi \cdot \rho$ est un concaténé proprement retourné de Π, ce qui est en contradiction avec la minimalité de $b_{queue}(\pi) - c_{queue}(\pi)$.

Si l'inversion ρ agit sur deux arêtes noires intrachromosomiques, alors $\pi \cdot \rho$ est un concaténé proprement retourné de $\Pi\rho$, d'où

$$
\begin{aligned}
& b(\Pi) - c(\Pi) + k(\Pi) + f(\Pi) \\
&= (b(\pi) - b_{queue}(\pi)) - (c(\pi) - c_{queue}(\pi)) + h(\pi) + f(\pi) \\
&= (b(\pi \cdot \rho) - b_{queue}(\pi \cdot \rho)) - (c(\pi \cdot \rho) - 1 - c_{queue}(\pi \cdot \rho)) + h(\pi \cdot \rho) + f(\pi \cdot \rho) \\
&= b(\Pi\rho) - c(\Pi\rho) + h(\Pi\rho) + f(\Pi\rho) + 1 \ .
\end{aligned}
$$

Comme on a $b(\Pi) - c(\Pi) + k(\Pi) + f(\Pi) > b(\Pi\rho) - c(\Pi\rho) + h(\Pi\rho) + f(\Pi\rho)$, le théorème est vrai pour le génome $\Pi\rho$. On en déduit : $d(\Pi) \leq d(\Pi\rho) + 1 = b(\Pi) - c(\Pi) + k(\Pi) + f(\Pi)$. ■

10.13 Coiffage de chromosomes

On s'intéresse désormais au cas général dans lequel les génomes Π et Γ peuvent avoir des ensembles de queues distincts et un nombre de chromosomes différent. Ci-dessous, on décrit un algorithme pour le calcul de $d(\Pi, \Gamma)$, dont la complexité est polynomiale en fonction du nombre de gènes mais exponentielle en fonction du nombre de chromosomes. Cet algorithme aboutit à l'algorithme de complexité (réellement) polynomiale décrit dans les sections suivantes.

Soient Π et Γ deux génomes ayant respectivement M et N chromosomes. Sans perte de généralité, on suppose $M \leq N$ et on étend Π avec $N - M$ chromosomes vides (figure 10.18). Par suite, $\Pi = \{\pi(1), \ldots, \pi(N)\}$ et $\Gamma = \{\gamma(1), \ldots, \gamma(N)\}$ contiennent le même nombre de chromosomes. Soit $\{coif_0, \ldots, coif_{2N-1}\}$ un ensemble de $2N$ entiers positifs distincts (appelés des *coiffes*) qui sont différents des gènes de Π (ou, de façon équivalente, de Γ). Soit $\hat{\Pi} = \{\hat{\pi}(1), \ldots, \hat{\pi}(N)\}$ un génome obtenu à partir de Π en ajoutant des coiffes aux extrémités de chaque chromosome, c'est-à-dire :

$$\hat{\pi}(i) = coif_{2(i-1)}, \pi(i)_1, \ldots, \pi(i)_{n_i}, coif_{2(i-1)+1}.$$

Toute inversion/translocation dans Π correspond à une inversion/translocation *interne* dans $\hat{\Pi}$. Si cette translocation est une fission, on suppose qu'il y a assez de chromosomes vides dans Π (l'exactitude de cette supposition sera éclaircie par la suite).

Tout tri de Π en Γ induit un tri de $\hat{\Pi}$ en un génome :

$$\hat{\Gamma} = \{\hat{\gamma}(1), \ldots, \hat{\gamma}(N)\}$$

(appelé un *coiffage* de Γ), avec :

$$\hat{\gamma}(i) = ((-1)^j coif_j, \hat{\gamma}(i)_1, \ldots, \hat{\gamma}(i)_{m_i}, (-1)^{k+1} coif_k)$$

pour $0 \leq j, k \leq 2N - 1$. Les génomes $\hat{\Pi}$ et $\hat{\Gamma}$ sont à queue commune ($\mathcal{T}(\hat{\Pi}) = \mathcal{T}(\hat{\Gamma}) = \bigcup_{i=0}^{2N-1} (-1)^i coif_i$). Il existe $(2N)!$ coiffages différents de Γ, chacun d'eux étant défini par la distribution de $2N$ coiffes de $\hat{\Pi}$ dans $\hat{\Gamma}$. On note $\mathbf{\Gamma}$ l'ensemble des $(2N)!$ coiffages de Γ. Le lemme suivant mène à un algorithme qui permet de calculer la distance génomique ; celle-ci est une fonction polynôme du nombre de gènes mais une fonction exponentielle du nombre de chromosomes N.

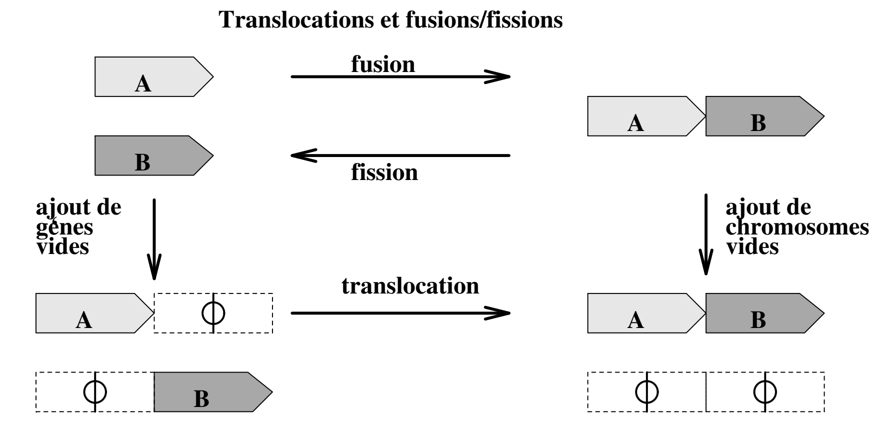


Figure 10.18 – Les fusions/fissions peuvent être imitées par des translocations en introduisant des chromosomes vides.

Lemme 10.29 *On a :* $d(\Pi, \Gamma) = \min_{\hat{\Gamma} \in \mathbf{\Gamma}} b(\hat{\Pi}, \hat{\Gamma}) - c(\hat{\Pi}, \hat{\Gamma}) + k(\hat{\Pi}, \hat{\Gamma}) + f(\hat{\Pi}, \hat{\Gamma})$.

Preuve La preuve découle du théorème 10.13 et de l'observation selon laquelle chaque tri de $\hat{\Pi}$ en un génome $\hat{\Gamma} \in \mathbf{\Gamma}$ par des inversions/translocations internes induit un tri de Π en Γ. ■

Soient $\hat{\pi}$ et $\hat{\gamma}$ des concaténés arbitraires de coiffages (ordonnés) $\hat{\Pi}$ et $\hat{\Gamma}$. Soit $G(\hat{\Pi}, \hat{\Gamma})$ un graphe obtenu à partir de $G(\hat{\pi}, \hat{\gamma})$ en supprimant toutes les queues (sommets de $G(\hat{\pi}, \hat{\gamma})$) du génome $\hat{\Pi}$ (ou, de façon équivalente, de $\hat{\Gamma}$) de $G(\hat{\pi}, \hat{\gamma})$. Des coiffages $\hat{\Gamma}$ différents correspondent à des graphes $G(\hat{\Pi}, \hat{\Gamma})$ différents. Le graphe $G(\hat{\Pi}, \hat{\Gamma})$ possède $2N$ sommets correspondant aux coiffes ; les arêtes grises incidentes à ces sommets définissent entièrement le coiffage $\hat{\Gamma}$. Par conséquent, la suppression de ces $2N$ arêtes grises transforme $G(\hat{\Pi}, \hat{\Gamma})$ en un graphe $G(\Pi, \Gamma)$ qui ne dépend pas du coiffage $\hat{\Gamma}$ (figure 10.19a, b, c et d).

Le graphe $G(\Pi, \Gamma)$ contient $2N$ sommets de degré 1 qui correspondent aux $2N$ coiffes de Π (appelées des Π-*coiffes*) et $2N$ sommets de degré 1 qui correspondent aux $2N$ queues de Γ (appelées des Γ-*queues*). Par conséquent, $G(\Pi, \Gamma)$ est une collection de cycles et de $2N$ chemins, chaque chemin commençant et finissant par une arête noire. Un chemin est appelé un $\Pi\Pi$-*chemin* (resp. un $\Gamma\Gamma$-*chemin*) s'il commence et finit par des Π-coiffes (resp. des Γ-queues) ; on dit que c'est un $\Pi\Gamma$-*chemin* s'il commence par une Π-coiffe et finit par une Γ-queue. Un sommet dans $G(\Pi, \Gamma)$ est un $\Pi\Gamma$-*sommet* si c'est une Π-coiffe dans un $\Pi\Gamma$-chemin et un $\Pi\Pi$-*sommet* s'il s'agit d'une Π-coiffe dans un $\Pi\Pi$-chemin. Les $\Gamma\Pi$- et $\Gamma\Gamma$-sommets sont définis de façon semblable (voir la figure 10.19d).

Tout coiffage $\hat{\Gamma}$ correspond à l'ajout de $2N$ arêtes grises dans le graphe $G(\Pi, \Gamma)$, chacune d'elles reliant une Π-coiffe à une Γ-queue. Ces arêtes transforment $G(\Pi, \Gamma)$ en $G(\hat{\Pi}, \hat{\Gamma})$, qui correspond à un coiffage $\hat{\Gamma}$ (figure 10.19e).

Par définition, $b(\Pi, \Gamma)$ est le nombre d'arêtes noires dans $G(\Pi, \Gamma)$ et $c(\Pi, \Gamma)$ le nombre total de cycles et de chemins dans $G(\Pi, \Gamma)$. Le paramètre $b(\Pi, \Gamma) = b(\hat{\Pi}, \hat{\Gamma})$ ne dépend pas du coiffage $\hat{\Gamma}$. Il est clair que $c(\hat{\Pi}, \hat{\Gamma})$ est inférieur ou égal à $c(\Pi, \Gamma)$ et qu'il y a égalité si chaque chemin dans $G(\Pi, \Gamma)$ est « fermé » par une arête grise dans $G(\hat{\Pi}, \hat{\Gamma})$. L'observation selon laquelle tout cycle dans $G(\hat{\Pi}, \hat{\Gamma})$ qui comporte un $\Pi\Pi$-chemin contient au moins un chemin de plus aboutit à l'inégalité $c(\hat{\Pi}, \hat{\Gamma}) \leq c(\Pi, \Gamma) - p(\Pi, \Gamma)$, où $p(\Pi, \Gamma)$ est le nombre de $\Pi\Pi$-chemins (ou, de façon équivalente, de $\Gamma\Gamma$-chemins) dans $G(\Pi, \Gamma)$.

On définit les notions de cycles/chemins chevauchants, de composantes orientées et non orientées, etc. dans le graphe $G(\Pi, \Gamma)$ de la manière habituelle, sans faire de distinction entre les cycles et les chemins dans $G(\Pi, \Gamma)$. On dit qu'un sommet π_j est *à l'intérieur* d'une composante U de π s'il vérifie $j \in [\bar{U}_{min}, \bar{U}_{max}]$. Une composante intrachromosomique pour les génomes Π et Γ est appelée une composante *réelle* si elle n'a aucune Π-coiffe et aucune Γ-queue à l'intérieur.

Pour des génomes Π et Γ, on définit $\mathcal{RU}(\Pi, \Gamma)$ comme étant l'ensemble des composantes réelles et $\mathcal{IU}(\Pi, \Gamma)$ comme l'ensemble des composantes intrachromosomiques (comme les définit le graphe $G(\Pi, \Gamma)$). On a clairement $\mathcal{RU}(\Pi, \Gamma) \subseteq \mathcal{IU}(\Pi, \Gamma)$. Les obstacles, les superobstacles et les forteresses pour l'ensemble $\mathcal{RU}(\Pi, \Gamma)$ sont appelés respectivement des *nœuds réels*, des *super-nœuds réels* et des *forteresses de nœuds réels*. Soit RK l'ensemble des nœuds réels (i.e. des obstacles pour l'ensemble $\mathcal{RU}(\Pi, \Gamma)$) et soit K l'ensemble des nœuds (i.e. des obstacles pour l'ensemble $\mathcal{IU}(\Pi, \Gamma)$). Un nœud de l'ensemble $K \setminus RK$ est un *semi-nœud* s'il ne contient pas de $\Pi\Pi$ ou de $\Gamma\Gamma$-sommet à l'intérieur. Il est clair que tout semi-nœud contient un $\Pi\Gamma$-chemin (sinon, ce serait un nœud réel). On note $r(\Pi, \Gamma)$ et $s(\Pi, \Gamma)$ le nombre de nœuds réels et de semi-nœuds pour les génomes Π et Γ, respectivement. On a clairement $k(\hat{\Pi}, \hat{\Gamma}) \geq r(\Pi, \Gamma)$, d'où

$$b(\hat{\Pi}, \hat{\Gamma}) - c(\hat{\Pi}, \hat{\Gamma}) + k(\hat{\Pi}, \hat{\Gamma}) \leq b(\Pi, \Gamma) - c(\Pi, \Gamma) + p(\Pi, \Gamma) + r(\Pi, \Gamma).$$

Cependant, cette borne n'est pas bien ajustée, car elle suppose qu'il existe un coiffage $\hat{\Gamma}$ qui maximise $c(\hat{\Pi}, \hat{\Gamma})$ et minimise $k(\hat{\Pi}, \hat{\Gamma})$ simultanément. En prenant en compte $s(\Pi, \Gamma)$, on aboutit à une meilleure borne pour la distance génomique, qui diffère d'au plus un réarrangement de cette distance.

10.14 Coiffes et queues

Les génomes Π et Γ sont dits *corrélés* si tous les nœuds réels dans $G(\Pi, \Gamma)$ sont situés sur le même chromosome et *non corrélés* sinon. Dans cette section, on restreint notre analyse à des génomes non corrélés (il s'avère que l'analyse de génomes corrélés implique des difficultés techniques supplémentaires) et on prouve une borne bien ajustée pour $d(\Pi, \Gamma)$ (celle-ci mène à une fonction plutôt compliquée utilisée dans la preuve du théorème de dualité) :

$$b(\Pi,\Gamma) - c(\Pi,\Gamma) + p(\Pi,\Gamma) + r(\Pi,\Gamma) + \lceil\frac{s(\Pi,\Gamma)}{2}\rceil \leq d(\Pi,\Gamma) \leq$$

$$b(\Pi,\Gamma) - c(\Pi,\Gamma) + p(\Pi,\Gamma) + r(\Pi,\Gamma) + \lceil\frac{s(\Pi,\Gamma)}{2}\rceil + 1$$

Les lemmes suivants suggèrent une façon de relier certains chemins dans $G(\Pi,\Gamma)$ par des arêtes orientées.

Lemme 10.30 *Pour tout* $\Pi\Pi$*-chemin et tout* $\Gamma\Gamma$*-chemin dans* $G(\Pi,\Gamma)$*, il existe une arête grise orientée ou interchromosomique qui relie ces chemins en un* $\Pi\Gamma$*-chemin.*

Lemme 10.31 *Pour toute paire de* $\Pi\Gamma$*-chemins non orientés situés sur le même chromosome, il existe une arête grise orientée qui relie ces chemins en un* $\Pi\Gamma$*-chemin.*

Lorsque l'on cherche un coiffage minimal, on commence par ignorer le terme $f(\hat{\Pi},\hat{\Gamma})$ dans le lemme 10.29 et l'on trouve un coiffage dont la distance génomique $d(\hat{\Pi},\hat{\Gamma})$ diffère d'au plus 1 de l'optimale. Le théorème suivant suggère une façon de trouver un tel coiffage « presque optimal » $\hat{\Gamma}$.

Théorème 10.14 *On a :* $\min_{\hat{\Gamma}\in\mathbf{\Gamma}} b(\hat{\Pi},\hat{\Gamma}) - c(\hat{\Pi},\hat{\Gamma}) + k(\hat{\Pi},\hat{\Gamma}) = b(\Pi,\Gamma) - c(\Pi,\Gamma) + p(\Pi,\Gamma) + r(\Pi,\Gamma) + \lceil\frac{s(\Pi,\Gamma)}{2}\rceil$.

Preuve Tout coiffage $\hat{\Gamma}$ définit une transformation de $G(\Pi,\Gamma)$ en $G(\hat{\Pi},\hat{\Gamma})$ en ajoutant consécutivement $2N$ arêtes grises à $G(\Pi,\Gamma)$: $G(\Pi,\Gamma) = G_0 \overset{g_1}{\to} G_1 \overset{g_2}{\to} \ldots \overset{g_{2N}}{\to} G_{2N} = G(\hat{\Pi},\hat{\Gamma})$. Pour un graphe G_i, les paramètres $b_i = b(G_i)$, $c_i = c(G_i)$, $p_i = p(G_i)$, $r_i = r(G_i)$ et $s_i = s(G_i)$ sont définis de la même façon que pour le graphe $G_0 = G(\Pi,\Gamma)$. Pour un paramètre ϕ, on définit $\Delta\phi_i$ comme étant $\phi_i - \phi_{i-1}$, i.e. $\Delta c_i = c_i - c_{i-1}$, etc. On note $\Delta_i = (c_i - p_i - r_i - \lceil\frac{s_i}{2}\rceil) - (c_{i-1} - p_{i-1} - r_{i-1} - \lceil\frac{s_{i-1}}{2}\rceil)$. Ci-dessous, on prouve que Δ_i est négatif pour $1 \leq i \leq 2N$, c'est-à-dire que l'ajout d'une arête grise n'augmente pas le paramètre $c(\Pi,\Gamma) - p(\Pi,\Gamma) - r(\Pi,\Gamma) - \lceil\frac{s(\Pi,\Gamma)}{2}\rceil$. Pour un i fixé, on ignore l'indice i, i.e. on note $\Delta = \Delta_i$, etc.

En fonction de l'arête g_i, les cas suivants sont possibles (l'analyse ci-dessous suppose que Π et Γ sont non-corrélés) :

Cas 1 : l'arête g_i *« ferme » un* $\Pi\Gamma$*-chemin* (autrement dit, g_i relie un $\Pi\Gamma$-sommet à un $\Gamma\Pi$-sommet à l'intérieur du même $\Pi\Gamma$-chemin). Si ce sommet est le seul $\Pi\Gamma$-sommet dans un semi-nœud, alors on a $\Delta c = 0, \Delta p = 0, \Delta r = 1$ et $\Delta s = -1$ (notons que ceci peut ne pas être vrai pour des génomes corrélés). Sinon, $\Delta c, \Delta p, \Delta r$ et Δs sont nuls. Dans les deux cas, on obtient $\Delta \leq 0$.

Cas 2 : l'arête g_i *relie un* $\Pi\Gamma$*-sommet à un* $\Gamma\Pi$*-sommet dans un* $\Pi\Gamma$*-chemin différent.* Cette arête « détruit » au plus deux semi-nœuds et on a $\Delta c = -1, \Delta p = 0, \Delta r = 0$ et $\Delta s \geq -2$. Donc Δ est négatif.

Cas 3 : l'arête g_i relie un $\Pi\Gamma$-sommet à un $\Gamma\Gamma$-sommet (ou un $\Gamma\Pi$-sommet à un $\Pi\Pi$-sommet). Cette arête « détruit » au plus un semi-nœud et on obtient $\Delta c = -1, \Delta p = 0, \Delta r = 0$et $\Delta s > -2$. Ceci implique $\Delta \leq 0$.

Cas 4 : l'arête g_i relie un $\Pi\Pi$-sommet à un $\Gamma\Gamma$-sommet. Cette arête ne peut détruire aucun semi-nœud et on a $\Delta c = -1, \Delta p = -1, \Delta r = 0$ et $\Delta s \geq 0$. D'où $\Delta \leq 0$.

Notons les égalités $b_{2N} = b(\hat{\Pi}, \hat{\Gamma}) = b(\Pi, \Gamma) = b_0$, $c_{2N} = c(\hat{\Pi}, \hat{\Gamma})$, $p_{2N} = 0$, $s_{2N} = 0$ et $r_{2N} = k(\hat{\Pi}, \hat{\Gamma})$. On en déduit donc $b(\hat{\Pi}, \hat{\Gamma}) - c(\hat{\Pi}, \hat{\Gamma}) + k(\hat{\Pi}, \hat{\Gamma}) = b_{2N} - c_{2N} + p_{2N} + r_{2N} + \lceil \frac{s_{2N}}{2} \rceil \geq b_0 - c_0 + p_0 + r_0 + \lceil \frac{s_0}{2} \rceil = b(\Pi, \Gamma) - c(\Pi, \Gamma) + p(\Pi, \Gamma) + r(\Pi, \Gamma) + \lceil \frac{s(\Pi,\Gamma)}{2} \rceil$.

Prouvons maintenant qu'il existe un coiffage $\hat{\Gamma}$ qui vérifie l'égalité suivante :

$$b(\hat{\Pi}, \hat{\Gamma}) - c(\hat{\Pi}, \hat{\Gamma}) + k(\hat{\Pi}, \hat{\Gamma}) = b(\Pi, \Gamma) - c(\Pi, \Gamma) + p(\Pi, \Gamma) + r(\Pi, \Gamma) + \lceil \frac{s(\Pi, \Gamma)}{2} \rceil.$$

Pour cela, on construit une suite de $2N$ arêtes grises $g_1, \ldots, g_{2N}$ qui relient les Π-coiffes aux Γ-queues dans $G(\Pi, \Gamma)$, de sorte que Δ_i soit nul pour tout $1 \leq i \leq 2N$.

Supposons que l'on ait déjà trouvé les $i-1$ premières arêtes ; soit G_{i-1} le résultat de l'ajout de ces $i-1$ arêtes dans $G(\Pi, \Gamma)$. Si G_{i-1} possède un $\Pi\Pi$-chemin, il contient également un $\Gamma\Gamma$-chemin et, d'après le lemme 10.30, il existe une arête grise orientée ou interchromosomique reliant ces chemins en un $\Pi\Gamma$-chemin orienté. On a clairement $\Delta c = -1$, $\Delta p = -1$, $\Delta r = 0$ et $\Delta s = 0$ pour cette arête, d'où $\Delta = 0$.

Si G_{i-1} possède au moins deux semi-nœuds (i.e. $s_{i-1} > 1$), soient v_1 et v_2 un $\Pi\Gamma$- et un $\Gamma\Pi$-sommet dans différents semi-nœuds. Si v_1 et v_2 sont dans des chromosomes différents de Π, alors l'arête grise $g_i = (v_1, v_2)$ « détruit » les deux semi-nœuds. On a donc $\Delta c = -1, \Delta p = 0, \Delta r = 0, \Delta s = -2$ et $\Delta = 0$. Si v_1 et v_2 appartiennent au même chromosome, alors, d'après le lemme 10.31, il existe une arête grise orientée qui relie ces chemins en un $\Pi\Gamma$-chemin orienté. Cette arête grise détruit deux semi-nœuds. On a donc également $\Delta = 0$ dans ce cas.

Si G_{i-1} possède le seul semi-nœud, on note P_1 un $\Pi\Gamma$-chemin situé dans celui-ci. Si c'est le seul $\Pi\Gamma$-chemin dans le semi-nœud, alors, pour une arête g_i « fermant » ce chemin, on a $\Delta c = 0, \Delta p = 0, \Delta r = 1$ et $\Delta s = -1$, d'où $\Delta = 0$. Sinon, $\Delta c, \Delta p, \Delta r$ et Δs sont nuls, d'où $\Delta = 0$.

Si G_{i-1} n'a aucun $\Pi\Pi$-chemin et aucun semi-nœud, on note g_i une arête fermant un $\Pi\Gamma$-chemin arbitraire dans G_{i-1}. Comme g_i n'appartient pas à un semi-nœud, $\Delta c, \Delta p, \Delta r, \Delta s$ et Δ sont nuls. Par conséquent, la suite d'arêtes construite $g_1, \ldots, g_{2N}$ transforme $G(\Pi, \Gamma)$ en $G(\hat{\Pi}, \hat{\Gamma})$ tel que l'on ait $b(\hat{\Pi}, \hat{\Gamma}) - c(\hat{\Pi}, \hat{\Gamma}) + k(\hat{\Pi}, \hat{\Gamma}) = b(\Pi, \Gamma) - c(\Pi, \Gamma) + p(\Pi, \Gamma) + r(\Pi, \Gamma) + \lceil \frac{s(\Pi,\Gamma)}{2} \rceil$. ■

Comme on a $0 \leq f(\Pi, \Gamma) \leq 1$, le lemme 10.29 et le théorème 10.14 impliquent que $b(\Pi, \Gamma) - c(\Pi, \Gamma) + p(\Pi, \Gamma) + r(\Pi, \Gamma) + \lceil \frac{s(\Pi,\Gamma)}{2} \rceil$ diffère d'au plus un réarrangement de la distance génomique $d(\Pi, \Gamma)$ pour les génomes non corrélés.

Dans la section suivante, on comble l'écart entre $b(\Pi,\Gamma) - c(\Pi,\Gamma) + p(\Pi,\Gamma) + r(\Pi,\Gamma) + \lceil \frac{s(\Pi,\Gamma)}{2} \rceil$ et $d(\Pi,\Gamma)$ pour des génomes arbitraires.

10.15 Théorème de dualité pour la distance génomique

La difficulté majeure pour combler l'écart entre $b(\Pi,\Gamma) - c(\Pi,\Gamma) + p(\Pi,\Gamma) + r(\Pi,\Gamma) + \lceil \frac{s(\Pi,\Gamma)}{2} \rceil$ et $d(\Pi,\Gamma)$ réside dans le « non recouvrement » des obstacles « restants » dans le théorème de dualité. Il s'avère que le théorème de dualité implique sept (!) paramètres, ce qui rend difficile à expliquer l'intuition qui se cache derrière. Le théorème 10.14 donne une indication pour les cinq premiers paramètres. Deux paramètres supplémentaires sont définis à présent.

Une composante de $G(\Pi,\Gamma)$ qui contient un $\Pi\Gamma$-chemin est dite *simple* si ce n'est pas un semi-nœud.

Lemme 10.32 *Il existe un coiffage optimal* $\hat{\Gamma}$ *qui ferme tous les* $\Pi\Gamma$*-chemins dans les composantes simples.*

Soit $\overline{G}$ un graphe obtenu à partir de $G(\Pi,\Gamma)$ en fermant tous les $\Pi\Gamma$-chemins situés dans des composantes simples. Sans provoquer de confusion, on peut utiliser les termes *nœuds réels, supernœuds réels* et *forteresses de nœuds réels* dans $\overline{G}$ et définir $rr(\Pi,\Gamma)$ comme étant le nombre de nœuds-réels dans $\overline{G}$. Notons que $rr(\Pi,\Gamma)$ ne coïncide pas nécessairement avec $r(\Pi,\Gamma)$.

Des génomes corrélés Π et Γ forment une *faible forteresse de nœuds réels* si (i) ils possèdent un nombre impair de nœuds réels dans $\overline{G}$, (ii) l'un des nœuds réels est le plus grand nœud réel dans $\overline{G}$, (iii) tout nœud réel sauf le plus grand est un supernœud réel dans $\overline{G}$ et (iv) $s(\Pi,\Gamma)$ est strictement positif. Notons qu'une faible forteresse de nœuds réels peut être transformée en une forteresse de nœuds réels en fermant les $\Pi\Gamma$-chemins contenus dans l'un des semi-nœuds. On définit deux paramètres supplémentaires de la façon suivante :

$$fr(\Pi,\Gamma) = \begin{cases} 1, & \text{si } \Pi \text{ et } \Gamma \text{ forment une forteresse de nœuds réels ou une faible forteresse de nœuds réels dans } \overline{G} \\ 0, & \text{sinon} \end{cases}$$

$$gr(\Pi,\Gamma) = \begin{cases} 1, & \text{s'il existe un plus grand nœud réel dans } \overline{G} \text{ et si } s(\Pi,\Gamma) \text{ est strictement positif} \\ 0, & \text{sinon} \end{cases}$$

Le théorème suivant prouve que *Tri_Génomique* (figure 10.20) résout le problème du tri génomique. La complexité temporelle de *Tri_Génomique* (dominée par la complexité temporelle du tri par inversions de permutations signées) est $O(n^4)$, où n désigne le nombre total de gènes.

Théorème 10.15 *(Hannenhalli et Pevzner, 1995[153])*

$$d(\Pi, \Gamma) =$$

$$b(\Pi, \Gamma) - c(\Pi, \Gamma) + p(\Pi, \Gamma) + rr(\Pi, \Gamma) + \lceil \frac{s(\Pi, \Gamma) - gr(\Pi, \Gamma) + fr(\Pi, \Gamma)}{2} \rceil.$$

10.16 Duplications génomiques

Le doublement du génome entier est un événement possible et que l'on a déjà observé. Il peut conduire à des avantages dans l'évolution, car un double génome possède deux copies de chaque gène, qui peuvent évoluer de façon indépendante. Comme les gènes peuvent développer de nouvelles fonctions, la duplication génomique peut aboutir à un progrès d'évolution rapide. Il existe des preuves selon lesquelles le génome vertébré a subi des duplications il y a deux cents millions d'années (Ohno *et al.*, 1968 [256]), avec des duplications plus récentes chez certaines familles de vertébrés (Postlethwait, 1998 [278]). Les cartes génétiques comparatives des génomes de plantes révèlent également de multiples duplications (Paterson *et al.*, 1996 [260]).

Le séquençage de la levure a révélé la preuve d'un ancien doublement de son génome il y a cent millions d'années (Wolfe et Shields, 1997 [369]). À l'origine, le génome dupliqué contenait deux copies identiques de chaque chromosome, mais au fur et à mesure des inversions, translocations, fusions et fissions, ces deux copies se sont désorganisées. El-Mabrouk *et al.*, 1999 [96] ont proposé une solution au problème de la reconstruction de l'ordre des gènes dans le génome de la levure avant le doublement.

Un génome dupliqué réarrangé contient deux copies de chaque gène. Le problème de la *duplication génomique* consiste à calculer le nombre minimal de translocations nécessaires pour transformer un génome dupliqué réarrangé en un *génome dupliqué parfait*, avec un nombre pair de chromosomes contenant deux copies identiques de chaque chromosome. Par exemple, un génome dupliqué réarrangé constitué de quatre chromosomes $\{abc, def, aef, dbc\}$ peut être transformé en un génome dupliqué parfait $\{abc, def, abc, def\}$ par une simple translocation des chromosomes aef et dbc. El-Mabrouk *et al.*, 1999 [96] ont proposé un algorithme polynomial pour le problème de la *duplication génomique*, dans le cas où les opérations de réarrangement ne sont que des translocations. L'algorithme utilise le théorème de dualité de Hannenhalli, 1995 [151] concernant la distance de translocation entre des génomes multichromosomiques. Le problème consistant à inventer une analyse des duplications génomiques plus adéquate avec à la fois des translocations et des inversions demeure non résolu.

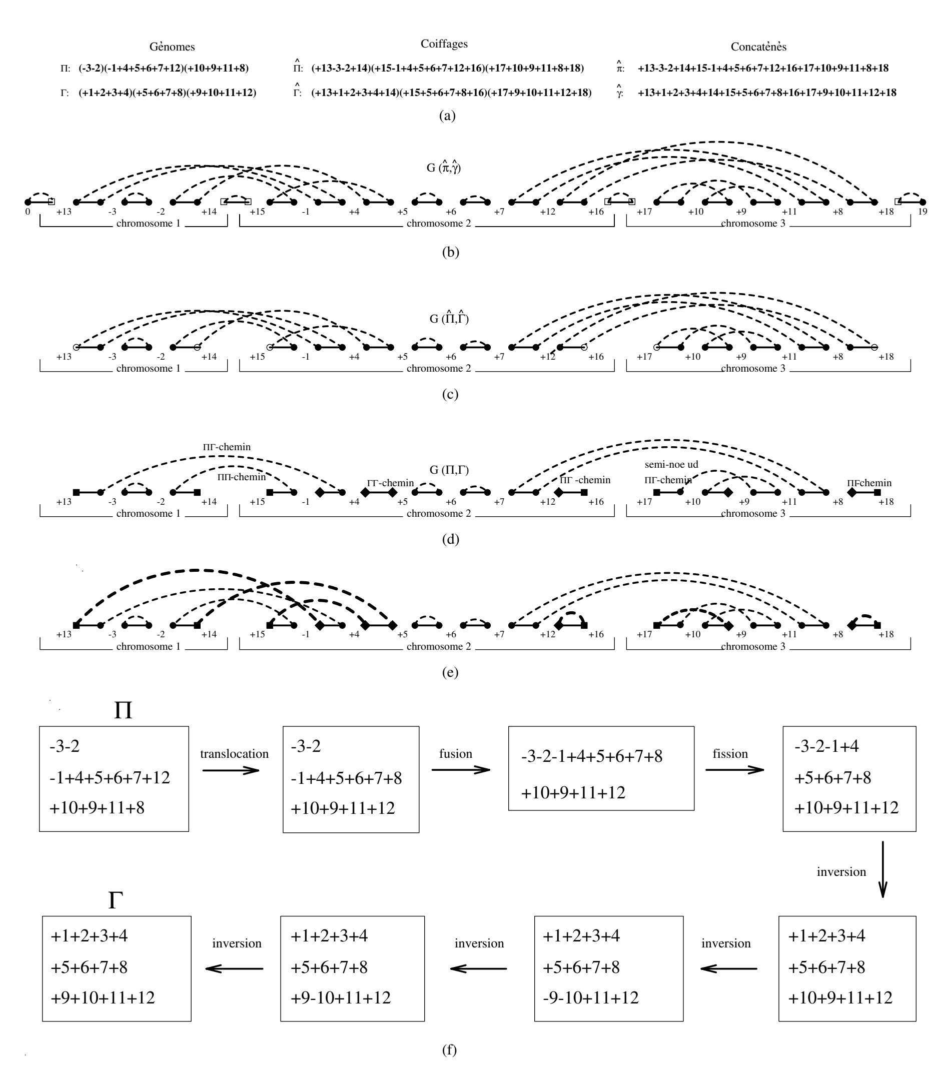


Figure 10.19 – (a) Les génomes Π et Γ, les coiffages $\hat{\Pi}$ et $\hat{\Gamma}$ et les concaténés $\hat{\pi}$ et $\hat{\gamma}$. (b) Le graphe $G(\hat{\pi}, \hat{\gamma})$. Les queues sont montrées sous la forme de carrés blancs. (c) Le graphe $G(\hat{\Pi}, \hat{\Gamma})$ est obtenu à partir de $G(\hat{\pi}, \hat{\gamma})$ en supprimant les queues. Les coiffes apparaissent sous la forme de cercles blancs. (d) Le graphe $G(\Pi, \Gamma)$ avec quatre cycles et six chemins ($c(\Pi, \Gamma) = 10$). Les Π-coiffes sont représentées par des carrés et les Γ-queues par des losanges. Pour les génomes Π et Γ, on a $b(\Pi, \Gamma) = 15$, $r(\Pi, \Gamma) = 0$, $p(\Pi, \Gamma) = 1$, $s(\Pi, \Gamma) = 1$ et $gr(\Pi, \Gamma) = fr(\Pi, \Gamma) = 0$; d'où $d(\Pi, \Gamma) = 15 - 10 + 1 + 0 + \lceil \frac{1-0+0}{2} \rceil = 7$. (e) Le graphe $G(\hat{\Pi}, \hat{\Gamma})$ correspond à un coiffage optimal de $\hat{\Gamma} = (+13+1+2+3+4-15)(-14+5+6+7+8+18)(+17+9+10+11+12+16)$. Les arêtes grises ajoutées sont représentées par des lignes épaisses en pointillé. (f) Le tri optimal de Π en Γ avec sept réarrangements.

Tri_Génomique (Π, Γ)

Construire le graphe $G = G(\Pi, \Gamma)$
Fermer tous les $\Pi\Gamma$-chemins dans les composantes simples de $G(\Pi, \Gamma)$ (lemme 10.32)
Fermer tous les $\Pi\Gamma$-chemins sauf un dans les composantes ayant plus d'un $\Pi\Gamma$-chemin à l'intérieur
tant que G contient un chemin
  s' il existe un $\Pi\Pi$-chemin dans G
    trouver une arête interchromosomique ou une arête orientée g reliant ce $\Pi\Pi$-chemin à un $\Gamma\Gamma$-chemin (lemme 10.30)
  sinon si G a plus de deux semi-nœuds
    trouver une arête interchromosomique ou une arête orientée g reliant les $\Pi\Gamma$-chemins dans toute paire de semi-nœuds (lemme 10.31)
  sinon si G a deux semi-nœuds
    si G a le plus grand nœud réel
      trouver une arête g fermant le $\Pi\Gamma$-chemin dans l'un de ces semi-nœuds
    sinon
      trouver une arête interchromosomique ou une arête orientée g reliant les $\Pi\Gamma$-chemins dans ces semi-nœuds (lemme 10.31)
  sinon si G a un semi-nœud
    trouver une arête g fermant le $\Pi\Gamma$-chemin dans ce semi-nœud
  sinon
    trouver une arête g fermant un $\Pi\Gamma$-chemin arbitraire
  ajouter une arête g au graphe G, i.e. $G \leftarrow G + \{g\}$
trouver un coiffage $\hat{\Gamma}$ défini par le graphe $G = G(\hat{\Pi}, \hat{\Gamma})$
trier le génome $\hat{\Pi}$ en $\hat{\Gamma}$ (théorème 10.14)
le tri de $\hat{\Pi}$ en $\hat{\Gamma}$ imite le tri de Π en Γ

Figure 10.20 – L'algorithme *Tri_Génomique*.

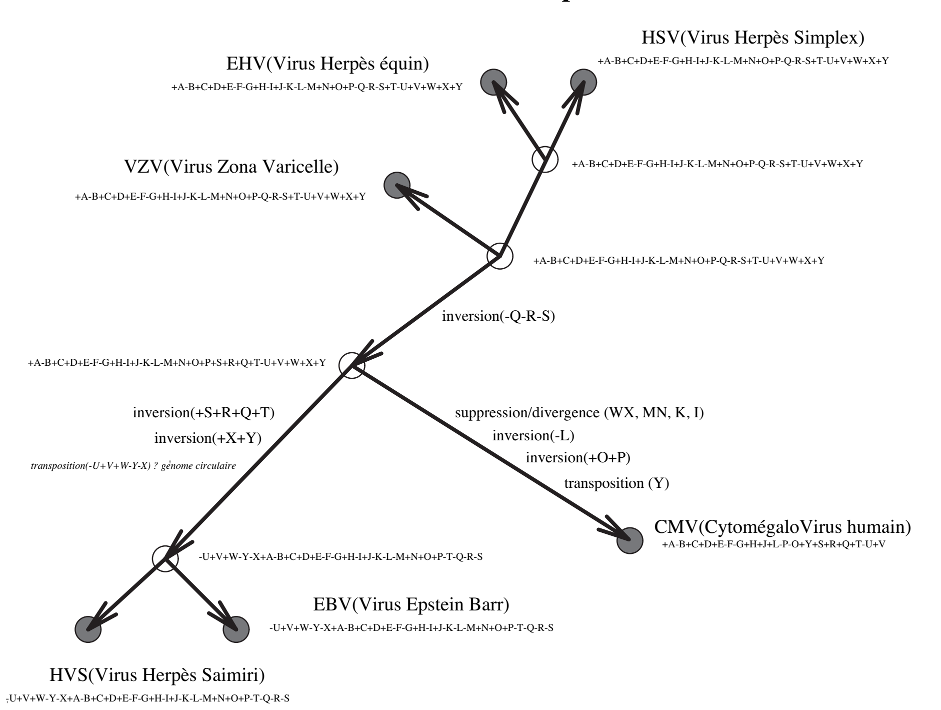


Figure 10.21 – Scénario putatif de l'évolution du virus de l'herpès.

10.17 Quelques autres problèmes et approches

10.17.1 Réarrangements génomiques et études phylogénétiques

Sankoff *et al.*, 1992 [304] ont innové en utilisant la distance de réarrangement pour les études sur l'évolution moléculaire. Une généralisation du problème de la distance génomique pour des génomes multiples correspond au problème suivant :

Problème de la distance génomique multiple Étant donné un ensemble de permutations $\pi^1, \ldots, \pi^k$, trouver une permutation σ telle que $\sum_{i=1,k} d(\pi^i, \sigma)$ soit minimale (d est la distance entre les génomes π^i et σ).

Dans le cas où $d(\pi, \sigma)$ est une distance d'inversion entre π et σ, on a montré que le problème de la distance génomique multiple était NP-complet (Caprara, 1999 [56]). De façon semblable à l'alignement multiple d'arbres d'évolution, il existe une généralisation du problème de la distance génomique multiple dans le cas où l'on ne connaît pas à l'avance l'arbre phylogénétique (figure 10.21). Comme la distance génomique multiple est complexe dans le cas de la distance d'inversion, la plupart des études sur l'évolution moléculaire génomique reposent sur la *distance des points de rupture*. Cette distance $d(\pi, \sigma)$ entre les permutations π et σ est définie comme étant le nombre de points de rupture dans $\pi\sigma^{-1}$. Bien que le problème de la distance génomique multiple soit également NP-complet avec cette formulation, Sankoff et Blanchette, 1998 [301] ont suggéré des heuristiques pour ce problème.

10.17.2 Algorithme rapide pour le tri par inversions

Berman et Hannenhalli, 1996 [33] et Kaplan *et al.*, 1997 [185] ont inventé des algorithmes rapides pour trier par inversions des permutations signées. L'algorithme quadratique de Kaplan *et al.*, 1997 [185] évite l'étape des transformations équivalentes de l'algorithme de Hannenhalli-Pevzner et explore les propriétés du graphe de chevauchements des *arêtes grises* (plutôt que de considérer le graphe de chevauchements des cycles). Ceci aboutit à une preuve plus élégante et plus simple du théorème 10.10.

Chapitre 11

Protéomique informatique

11.1 Introduction

En quelques secondes, un spectromètre de masse est capable de casser un peptide en fragments et de mesurer leur masse (le spectre du peptide). Le problème du séquençage peptidique consiste à trouver la séquence d'un peptide à partir de son spectre. Pour un processus de fragmentation idéal (chaque peptide est fendu entre toutes les paires d'acides aminés consécutifs) et un spectromètre de masse idéal, le problème du séquençage peptidique est simple. En pratique, les processus de fragmentation sont loin d'être idéaux, rendant difficile le séquençage de peptides *de novo*.

La recherche dans des bases de données est une alternative au séquençage peptidique *de novo* et la spectrométrie de masse est très efficace dans l'identification de protéines déjà présentes dans les bases de données génomiques (Patterson et Aebersold, 1995 [261]). En spectrométrie de masse, la recherche dans des bases de données (Mann et Wilm, 1994 [230], Eng *et al.*, 1994 [97], Taylor et Johnson, 1997 [335] et Fenyo *et al.*, 1998 [101]) repose sur la capacité à « chercher la réponse à la fin du livre », lors de l'étude de génomes d'organismes largement séquencés. Un spectre expérimental peut être comparé à des spectres théoriques pour chaque peptide d'une base de données ; celui qui s'ajuste le mieux fournit habituellement la séquence du peptide expérimental. En particulier, Eng *et al.*, 1994 [97] ont identifié des protéines du complexe MHC de classe II, tandis que Clauser *et al.*, 1999 [72] ont identifié des protéines liées aux effets de la prééclampsie. Cependant, compte tenu du caractère dynamique des échantillons introduits dans un spectromètre de masse et des multiples modifications et mutations possibles, la fiabilité des méthodes de recherche dans les bases de données (qui reposent sur des appariements précis ou presque) peut être mise en question. Les algorithmes *de novo* qui tentent d'interpréter les spectres de masse en tandem en l'absence de base de données sont inestimables pour l'identification de protéines inconnues, mais ils sont surtout utiles pour travailler avec des spectres de haute qualité.

Comme les protéines sont des parties de systèmes complexes de signalement cellulaire et de régulation métabolique, elles sont sujettes à un nombre presque incalculable de modifications biologiques (telles que la phosphorylation et la glycosylation) et de variations génétiques (Gooley et Packer, 1997 [134] et Krishna et Wold, 1998 [207]). Par exemple, il existe au moins 1000 kinases dans le génome humain, ce qui indique que la phosphorylation est un mécanisme fréquent pour la transmission de signaux et l'activation d'enzymes. Presque toutes les séquences protéiques sont modifiées post-translationellemment et on connaît environ 200 types de modifications de résidus d'acides aminés. Comme les modifications post-translationelles ne peuvent généralement pas être déduites des séquences d'ADN, les trouver restera un problème ouvert, même après complétion du génome humain. Ceci fait naître également un problème informatique de taille pour l'ère post-génomique : étant donnée une très grande collection de spectres représentant le protéome humain, trouver lesquels des 200 types de modifications sont présents dans chaque gène humain.

Depuis l'article classique de Biemann et Scoble, 1987 [35], il y a eu plusieurs succès dans l'identification de protéines modifiées par spectrométrie de masse. L'analyse informatique de peptides modifiés a été introduite par Mann et Wilm, 1994 [230] et Yates *et al.*, 1995 [374], [373]. Le problème est particulièrement important car les techniques de spectrométrie de masse introduisent parfois des modifications chimiques dans les peptides indigènes et les rendent « invisibles » pour les logiciels de recherche dans des bases de données. Mann et Wilm, 1994 [230] ont utilisé une combinaison d'algorithme *de novo* partiel et de recherche sur bases de données dans leur approche d'*étiquette peptidique* (PST). Une étiquette peptidique est une courte sous-chaîne clairement identifiable d'un peptide, qui est utilisée pour réduire la recherche aux peptides qui la contiennent. Yates *et al.*, 1995 [374] ont suggéré une approche de recherche exhaustive qui produit (implicitement) une base de données virtuelle de tous les peptides modifiés pour un petit ensemble de modifications et qui apparie le spectre avec la base de données virtuelle. On aboutit à un gros problème combinatoire, même pour un petit ensemble de types de modifications. Une autre limite est que des modifications très encombrantes comme la glycosylation désorganisent le modèle de fragmentation et ne seraient pas pratiques à analyser par cette méthode.

En spectrométrie de masse, la recherche sur bases de données tolérant les mutations peut être formulée de la façon suivante : étant donné un spectre expérimental, trouver le peptide qui s'apparie le mieux à lui parmi les peptides qui sont à moins de k mutations d'un peptide de la base de données. Ce problème est loin d'être simple, car des peptides très similaires peuvent avoir des spectres très différents. Pevzner *et al.*, 2000 [270] ont introduit la notion de similitude spectrale, qui mène à un algorithme qui identifie les spectres voisins, même si les peptides correspondants ont des modifications ou des mutations multiples. L'algorithme révèle des modifications peptidiques possibles sans recherche exhaustive ; il ne nécessite donc pas la production d'une base de données virtuelle de peptides modifiés.

Bien que la recherche sur base de données soit très utile, un biologiste qui tente de cloner un *nouveau* gène en se basant sur les données de spectrométrie de masse a besoin de *de novo* plutôt que d'algorithmes d'appariement avec base de données. Cependant, jusqu'à une date récente, le séquençage par spectrométrie de masse n'était pas très pratiqué et avait un impact limité sur la découverte de nouvelles protéines. Il existe quelques précieux exemples de clonage d'un gène basé uniquement sur l'information de séquences trouvées par spectrométrie de masse (Lingner *et al.*, 1997 [223]).

Le récent progrès en séquençage peptidique *de novo*, associé à l'acquisition de données de spectrométrie de masse automatisée, peut ouvrir la porte au « séquençage protéomique ». De longues bandes de séquences protéiques pourraient être assemblées en suivant la production et le séquençage d'ensembles chevauchants de peptides provenant du traitement de mixtures protéiques avec des enzymes protéolytiques de spécificité différente. La détermination de séquences protéiques complètes a déjà été mise en œuvre avec une telle stratégie sur une seule protéine (Hopper *et al.*, 1989 [166]).

11.2 Le problème du séquençage peptidique

Soit A l'ensemble des acides aminés de masse moléculaire $m(a)$ ($a \in A$). Un *peptide* $P = p_1, \ldots, p_n$ est une séquence d'acides aminés et la masse (parente) du peptide P est $m(P) = \sum m(p_i)$. Un *peptide partiel* P' est une sous-chaîne $p_i \ldots p_j$ de P de masse $\sum_{i \leq t \leq j} m(p_t)$.

La fragmentation de peptides dans un *spectromètre de masse en tandem* peut être caractérisée par un ensemble de nombres $\Delta = \{\delta_1, \ldots, \delta_k\}$ représentant des *ion-types*. Un *δ-ion* d'un peptide partiel $P' \subset P$ est une modification de P' qui a pour masse $m(P') - \delta$. Pour la spectrométrie de masse en tandem, le spectre *théorique* du peptide P peut être calculé en soustrayant tous les ion-types possibles $\delta_1, \ldots, \delta_k$ des masses de tous les peptides partiels de P (tout peptide partiel engendre k masses dans le spectre théorique). Un spectre (expérimental) $S = \{s_1, \ldots, s_m\}$ est un ensemble de masses d'ions (fragments). L'*appariement* entre le spectre S et le peptide P est le nombre de masses communes au spectre expérimental et au spectre théorique (dénombrement de pics communs). Dancik *et al.*, 1999 [79] ont formulé le problème suivant :

Problème du séquençage peptidique Étant donnés un spectre S, l'ensemble des ion-types Δ et la masse m, trouver un peptide de masse m présentant l'appariement maximal pour le spectre S.

On note P_i le peptide partiel *N-terminal* $p_1, \ldots, p_i$ et P_i^- le peptide *C-terminal* $p_{i+1}, \ldots, p_n$, pour $i = 1, \ldots, n$. En pratique, un spectre obtenu par spectrométrie de masse en tandem (MS/MS pour *Mass-Spectrometry*) est principalement constitué de certains δ-ions de peptides partiels N-terminaux et C-terminaux. Pour refléter cela, un spectre théorique MS/MS n'est constitué

que d'ions de peptides N-terminaux et C-terminaux (figure 11.1). Par exemple, les ions N-terminaux les plus fréquents sont habituellement les b-ions (b_i correspond à P_i avec $\delta = -1$) et les ions C-terminaux les plus courants sont normalement les y-ions (y_i correspond à P_i^- avec $\delta = 19$). D'autres ions N-terminaux fréquents pour un spectromètre de masse piège à ions (a, bH_2O et $b\text{–}NH_3$) sont présentés en figure 11.2. Par ailleurs, au lieu du dénombrement de pics communs, la recherche sur bases de données existantes et les algorithmes *de novo* utilisent des fonctions objectives plus sophistiquées (telles que le dénombrement de pics communs pondéré).

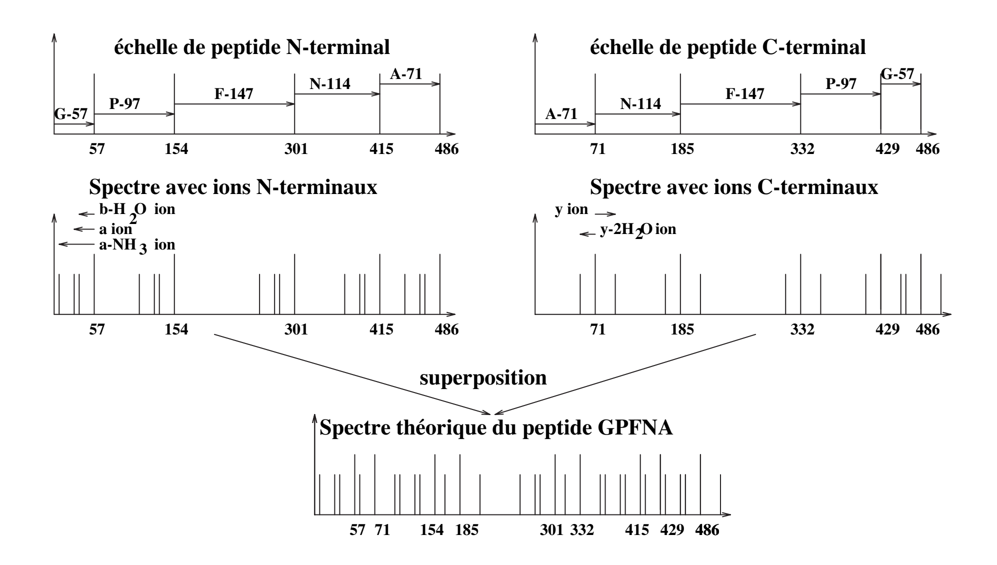


Figure 11.1 – Spectre théorique MS/MS du peptide GPFNA de masse parente $57 + 97 + 147 + 114 + 71 = 486$.

11.3 Graphes spectraux

Les algorithmes de développement du séquençage peptidique font appel soit à la recherche exhaustive, soit aux graphes spectraux. La première de ces deux approches (Sakurai *et al.*, 1984 [294]) implique la production de toutes les séquences d'acides aminés et des spectres théoriques correspondants. Le but est de trouver une séquence présentant le meilleur appariement possible entre le spectre expérimental et le spectre théorique. Comme le nombre de séquences croît exponentiellement avec la longueur du peptide, différentes techniques d'élagage ont été inventées pour limiter l'explosion combinatoire dans les mé-

thodes globales. L'élagage par préfixes (Hamm *et al.*, 1986 [149], Johnson et Biemann, 1989 [182], Zidarov *et al.*, 1990 [378] et Yates *et al.*, 1991 [375]) restreint l'espace calculatoire aux séquences dont les préfixes s'apparient bien au spectre expérimental. La difficulté avec cette approche est que l'élagage écarte fréquemment la séquence correcte si ses préfixes sont faiblement représentés dans le spectre. Un autre problème est que l'information spectrale n'est utilisée qu'*après* la production des séquences peptidiques possibles.

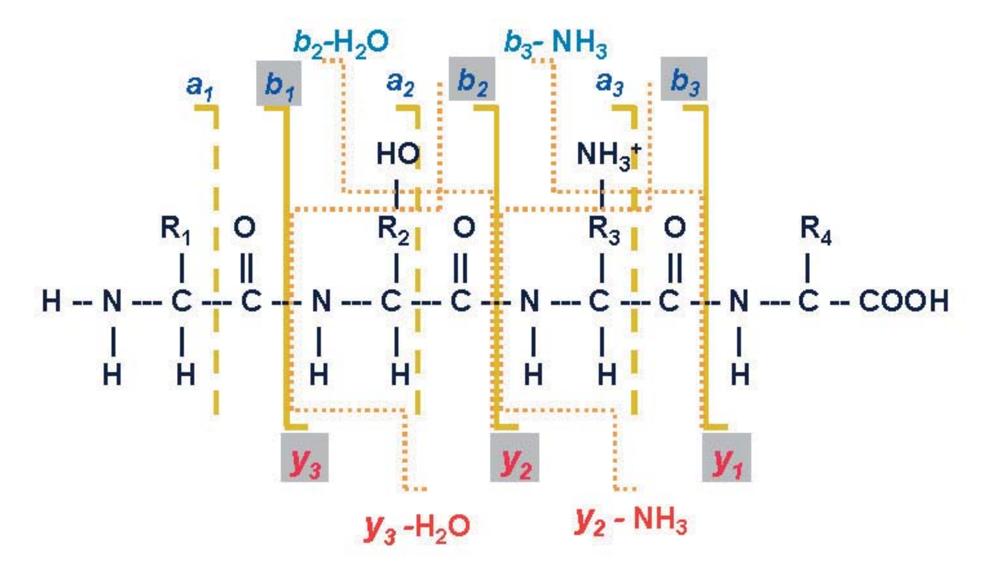


Figure 11.2 – Modèles de fragmentation typique dans la spectrométrie de masse en tandem.

Les approches fondées sur les graphes spectraux ont tendance à être plus efficaces car elle utilisent l'information spectrale *avant* d'évaluer toute séquence candidate. Dans cette approche, les pics d'un spectre sont transformés en un *graphe spectral* (Bartels, 1990 [26], Fernández-de-Cossío *et al.*, 1995 [103], Taylor et Johnson, 1997 [335] et Dancik *et al.*, 1999 [79]). Les pics du spectre sont les sommets du graphe spectral ; deux sommets sont reliés par une arête si leur masse diffère d'un acide aminé. Chaque pic d'un spectre expérimental est transformé en plusieurs sommets dans un graphe spectral ; chaque sommet représente une association possible avec un fragment ion-type pour le pic. Le problème du séquençage peptidique revient donc à trouver le plus long chemin dans le graphe acyclique orienté obtenu. Comme on connaît des algorithmes efficaces pour trouver les chemins les plus longs dans des graphes acycliques orientés (Cormen *et al.*, 1989 [75]), de telles approches ont la capacité d'élaguer efficacement l'ensemble de tous les peptides en l'ensemble des chemins de plus haut score dans le graphe spectral.

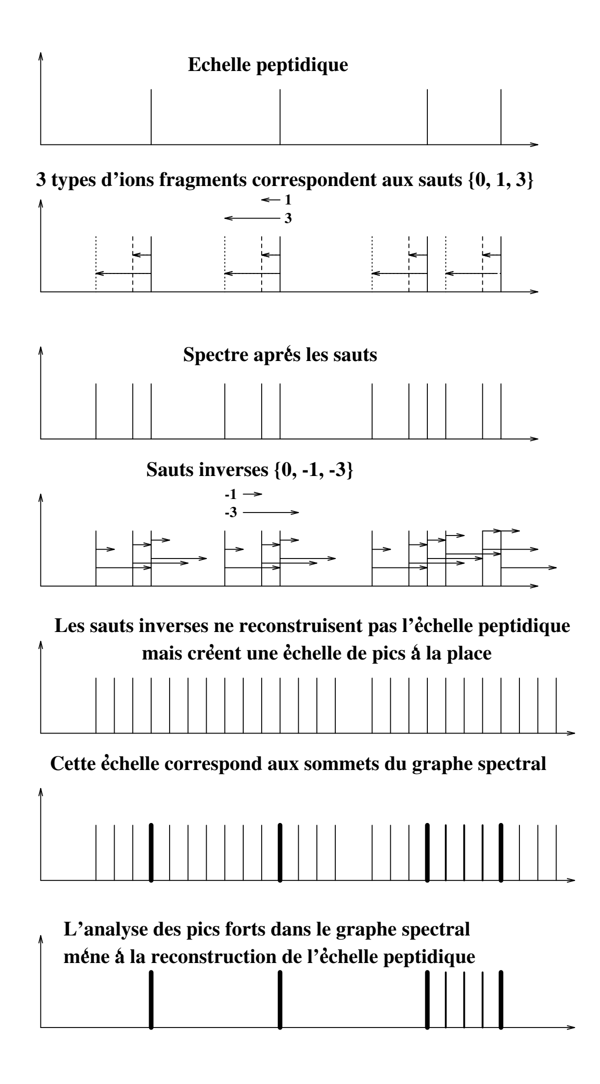


Figure 11.3 – Les sauts inverses créent un ensemble de sommets dans le graphe spectral. Les « pics forts » (les bâtons dessinés en gras) correspondent à des pics obtenus par des sauts inverses multiples.

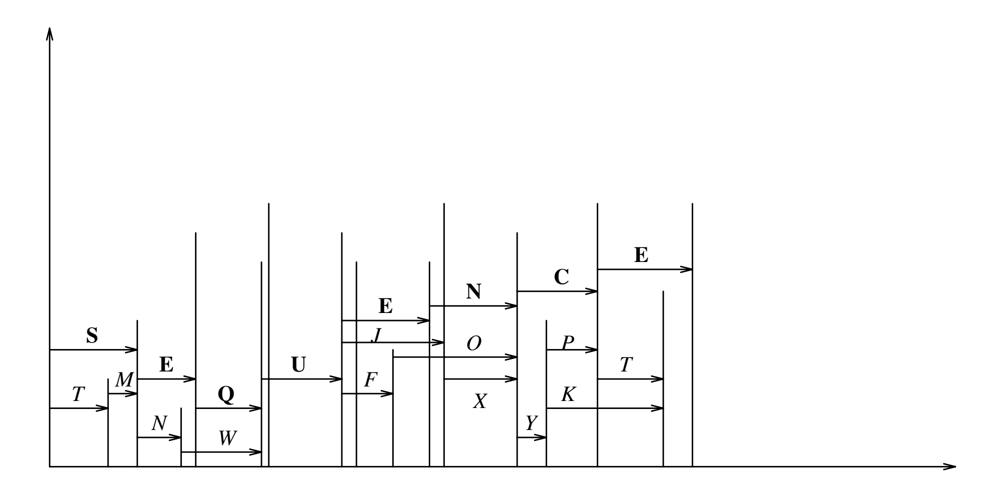


Figure 11.4 – Chemins multiples dans un graphe spectral.

L'approche fondée sur le graphe spectral est illustrée en figure 11.3. Comme les pics significatifs dans le spectre sont créés par une échelle peptidique (masses des peptides partiels) par des δ-sauts, on peut penser que des δ-sauts inverses vont reconstruire le spectre idéal, aboutissant ainsi au séquençage peptidique. La figure 11.3 montre que ceci n'est pas vrai et que l'on a besoin d'une analyse plus prudente des δ-sauts inverses (fondée sur la notion de graphe spectral).

Dans un souci de simplicité, supposons qu'un spectre MS/MS $S = \{s_1, \ldots, s_m\}$ soit principalement constitué d'ions N-terminaux et que nous le transformions en un graphe spectral $G_\Delta(S)$ (Bartels, 1990 [26]). Les sommets du graphe sont les entiers $s_i + \delta_j$; ils représentent les masses potentielles des peptides partiels. Chaque pic du spectre $s \in S$ produit k sommets $V(s) = \{s + \delta_1, \ldots, s + \delta_k\}$. L'ensemble des sommets d'un graphe spectral est donc $\{s_{initial}\} \cup V(s_1) \cup \cdots \cup V(s_m) \cup \{s_{final}\}$, avec $s_{initial} = 0$ et $s_{final} = m(P)$. Deux sommets u et v sont reliés par un arc allant de u à v si $v - u$ est la masse d'un acide aminé ; l'arc est alors *étiqueté* par cet acide aminé. Si l'on regarde les sommets comme des peptides N-terminaux potentiels, l'arête allant de u à v implique que la séquence en v peut être obtenue en étendant la séquence en u par un acide aminé (figures 11.4 et 11.5).

Un spectre S d'un peptide P est dit *complet* si S contient au moins un ion-type correspondant à P_i, pour tout $1 \leq i \leq n$. L'utilisation d'un graphe spectral est fondée sur l'observation suivante : pour un spectre complet, il existe un chemin de longueur n, allant de $s_{initial}$ à s_{final} dans $G_\Delta(S)$, et qui est étiqueté P. D'après cette observation, le problème du séquençage peptidique par spectrométrie de masse en tandem revient à trouver le chemin correct dans l'ensemble de tous les chemins.

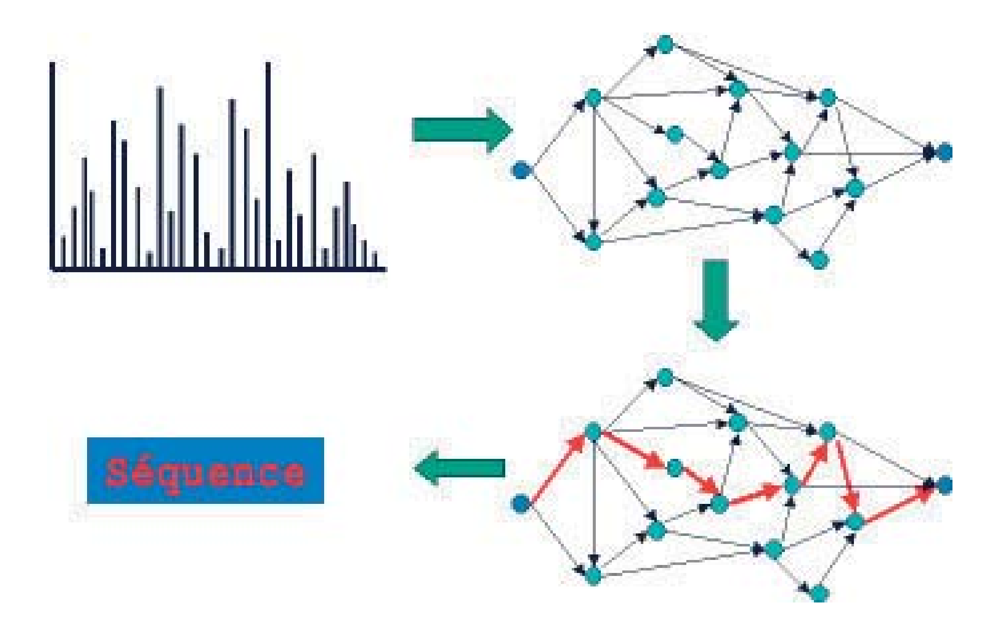
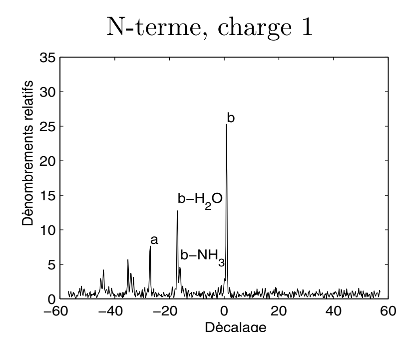


Figure 11.5 – Les parasites d'un spectre produisent de nombreux sommets et arêtes « erronés » dans le graphe spectral et déguisent les arêtes correspondant à la vraie séquence peptidique. La reconstruction d'une séquence revient à trouver un chemin optimal dans le graphe spectral.

Malheureusement, les spectres expérimentaux sont fréquemment incomplets. Par ailleurs, les expériences MS/MS réalisées avec le même peptide mais avec un type de spectromètre de masse différent vont produire des données sensiblement différentes. Des méthodes d'ionisation différentes ont un impact considérable sur les tendances à produire des ion-types de fragments particuliers. Par conséquent, tout algorithme de séquençage peptidique devrait être ajusté à un type particulier de spectromètre de masse. Pour traiter ce problème, Dancik *et al.*, 1999 [79] ont décrit un algorithme pour un apprentissage *automatique* des ion-types à partir d'un échantillon de spectres expérimentaux de séquences connues. Ils ont introduit la *fonction de fréquence de décalage*, qui évalue les tendances ion-types pour des spectromètres de masse particuliers et mène à un algorithme de séquençage peptidique indépendant du choix des instruments.

11.4 Apprentissage d'ion-types

Si les ion-types $\Delta = \{\delta_1, \ldots, \delta_k\}$ produits par un spectromètre de masse donné ne sont pas connus, le spectre ne peut être interprété. Ci-dessous, on



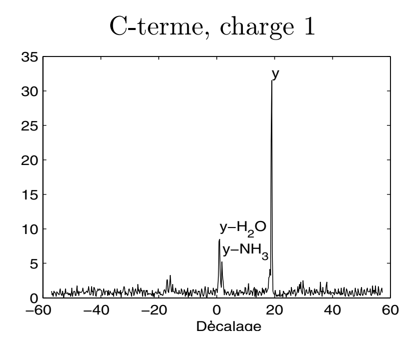


Figure 11.6 – Fonction de fréquence de décalage pour un peptide N-terminal (à gauche) et un peptide C-terminal (à droite). Les axes horizontaux représentent les décalages entre les pics dans les spectres et les masses des molécules peptidiques partielles. Les axes verticaux représentent les dénombrements de décalages normalisés, avec 1 comme dénombrement moyen.

montre comment apprendre l'ensemble Δ et les caractéristiques d'un échantillon de spectres expérimentaux de séquences connues *sans* connaissance préalable des modèles de fragmentation.

Soit $S = \{s_1, \ldots, s_m\}$ un spectre correspondant au peptide P. Un peptide partiel P_i et un pic s_j ont un décalage $x_{ij} = m(P_i) - s_j$; on peut traiter x_{ij} comme une variable aléatoire. Comme la probabilité des décalages correspondant à de « vrais » ions de fragments est nettement supérieure à celle de décalages aléatoires, les pics dans la distribution empirique des décalages révèlent des ions de fragments. Les statistiques des décalages sur tous les ions et tous les peptides partiels fournissent un algorithme d'apprentissage fiable pour les ion-types (Dancik *et al.*, 1999 [79]).

Étant donnés le spectre S, le décalage x et la précision ε, soit $H(x, S)$ le nombre de paires (P_i, s_j), avec $i = 1, \ldots, n-1$ et $j = 1, \ldots, m$, qui ont le décalage $m(P_i) - s_j$ et qui sont situées à une distance inférieure à ε de x. La *fonction de fréquence de décalage* est définie comme étant $H(x) = \sum_S H(x, S)$, où l'on somme sur tous les spectres de l'échantillon d'apprentissage (figure11.6). Pour apprendre de l'information sur les ions C-terminaux, on fait la même chose pour les paires (P_i^-, s_j). Les décalages $\Delta = \{\delta_1, \ldots, \delta_k\}$ correspondant aux pics de $H(x)$ représentent les ion-types produits par un spectromètre de masse donné.

Les pics d'un spectre diffèrent en *intensité* et il faut fixer un seuil pour distinguer le signal des parasites dans un spectre, avant de le transformer en graphe spectral. Des seuils faibles mènent à une croissance excessive du graphe spectral, tandis que des seuils hauts aboutissent à la fragmentation du graphe spectral. La fonction de fréquence de décalage permet d'établir les seuils d'intensité de façon rigoureuse (Dancik *et al.*, 1999 [79]).

11.5 Score des chemins dans les graphes spectraux

Le but du score est de quantifier à quel point un peptide candidat « explique » un spectre et de choisir le peptide qui explique le mieux le spectre considéré. Si $p(P, S)$ est la probabilité qu'un peptide P produise le spectre S, le but est de trouver un peptide P qui maximise $p(P, S)$ pour un spectre S donné. Ci-dessous, on décrit un modèle probabiliste, on évalue $p(P, S)$ et on en déduit un schéma de score pour les chemins dans le graphe spectral. Le plus long chemin dans le graphe spectral pondéré correspond au peptide P qui explique le mieux le spectre S.

Dans une approche probabiliste, la spectrométrie de masse en tandem est caractérisée par un ensemble d'ion-types $\Delta = \{\delta_1, \ldots, \delta_k\}$ et leur probabilité $\{p(\delta_1), \ldots, p(\delta_k)\}$, de sorte que les δ_i-ions d'un peptide partiel soient produits indépendamment avec la probabilité $p(\delta_i)$. Un spectromètre de masse produit également un parasite aléatoire (chimique ou électronique), qui peut engendrer un pic en chaque position avec la probabilité q_R. Par conséquent, un pic situé à une position qui correspond à un δ_i-ion est produit avec la probabilité $q_i = p(\delta_i) + (1 - p(\delta_i))q_R$, que l'on peut estimer à partir des distributions empiriques. Un peptide partiel peut théoriquement avoir jusqu'à k pics dans le spectre. La probabilité qu'il en possède k est de $\prod_{i=1}^{k} q_i$ et celle qu'il n'en possède aucun est de $\prod_{i=1}^{k}(1 - q_i)$. Le modèle probabiliste définit la probabilité $p(P, S)$ qu'un peptide P produise un spectre S.

Supposons qu'un peptide partiel candidat P_i produise les ions $\delta_1, \ldots, \delta_l$ (les ions « présents ») et pas les ions $\delta_{l+1}, \ldots, \delta_k$ (les ions « manquants ») dans le spectre S. Les l ions présents vont aboutir à un sommet dans le graphe spectral qui correspond à P_i. Comment peut-on attribuer un score à ce sommet ? Une approche naïve consisterait à attribuer une prime pour les ions expliqués, qui suggère que le score de ce sommet devrait être proportionnel à $q_1 \cdots q_l$ ou peut-être à $\frac{q_1}{q_R} \cdots \frac{q_l}{q_R}$ (pour normaliser les probabilités par rapport au parasite). Une telle approche est insuffisante et, pour obtenir une amélioration significative, on doit accorder une prime pour les ions présents et attribuer une pénalité pour ceux qui manquent. Le score (de probabilité) du sommet est donc de la forme :

$$\frac{q_1}{q_R} \cdots \frac{q_l}{q_R} \frac{(1 - q_{l+1})}{(1 - q_R)} \cdots \frac{(1 - q_k)}{(1 - q_R)}.$$

Expliquons le rôle de ce principe pour la résolution d'une simple alternative entre le dipeptide GG et l'acide aminé N de même masse. En l'absence de pénalité pour les ions manquants, GG est sélectionné sur N en présence de *tout* pic supportant la position du premier G (même un pic très faible correspondant au parasite aléatoire). Une telle règle aboutit à de nombreuses fausses interprétations GG-abondantes et indique qu'une meilleure règle serait de voter pour GG s'il est supporté par des ion-types majeurs avec des intensités suffisantes ;

ceci est automatiquement appliqué par le score « prime pour les ions présents, pénalité pour les manquants ».

Pour simplifier, on suppose que tous les peptides partiels sont équiprobables et on ignore les intensités des pics. On discrétise l'espace de toutes les masses dans l'intervalle allant de 0 à la masse parente $m(P) = M$, on note $T = \{0, \ldots, M\}$ et on représente le spectre sous la forme d'un vecteur $S = \{s_1, \ldots, s_M\}$ de dimension M tel que s_t soit l'indicateur de présence ou d'absence de pics en position t ($s_t = 1$ s'il y a un pic en position t et $s_t = 0$ sinon). Pour un peptide P donné et une position t, s_t est une variable aléatoire 0-1 de distribution de probabilité $p(P, s_t)$.

Soit $T_i = \{t_{i1}, \ldots, t_{ik}\}$ l'ensemble des positions qui représentent les Δ-ions d'un peptide partiel P_i, avec $\Delta = \{\delta_1, \ldots, \delta_k\}$. Soit $R = T \setminus \bigcup_i T_i$ l'ensemble des positions qui ne sont associées à aucun peptide partiel. La distribution de probabilité $p(P, s_t)$ dépend du fait que t soit dans T_i ou dans R. Pour une position $t = t_{ij} \in T_i$, la probabilité $p(P, s_t)$ est donnée par

$$p(P, s_t) = \begin{cases} q_j \,, & \text{si } s_t = 1 \text{ (un pic en position } t\text{)}\,; \\ 1 - q_j \,, & \text{sinon.} \end{cases} \tag{11.1}$$

De la même façon, pour $t \in R$, la probabilité $p(P, s_t)$ est donnée par

$$p_R(P, s_t) = \begin{cases} q_R \,, & \text{si } s_t = 1 \text{ (parasite aléatoire en position } t\text{)}\,; \\ 1 - q_R \,, & \text{sinon.} \end{cases} \tag{11.2}$$

La probabilité globale des pics « parasites » dans le spectre est $\prod_{t \in R} p_R(P, s_t)$.

Soit $p(P_i, S) = \prod_{t \in T_i} p(P, s_t)$ la probabilité qu'un peptide P_i produise un spectre donné aux positions de l'ensemble T_i (on ignore toutes les autres positions). Dans un souci de simplicité, on suppose que chaque pic du spectre n'appartient qu'à un ensemble T_i et que toutes les positions sont indépendantes. On a alors

$$p(P, S) = \prod_{t=1}^{M} p(P, s_t) = \left(\prod_{i=1}^{n} p(P_i, S) \right) \prod_{t \in R} p_R(P, s_t).$$

Pour un spectre S donné, la valeur $\prod_{t \in T} p_R(P, s_t)$ ne dépend pas de P et maximiser $p(P, S)$ revient à maximiser :

$$\frac{p(P, S)}{p_R(S)} = \frac{\prod_{i=1}^{n} \prod_{j=1}^{k} p(P, s_{t_{ij}}) \prod_{t \in R} p_R(P, s_t)}{\prod_{t \in T} p_R(P, s_t)} = \prod_{i=1}^{n} \prod_{j=1}^{k} \frac{p(P, s_{t_{ij}})}{p_R(P, s_{t_{ij}})},$$

où $p_R(S) = \prod_{t \in T} p_R(P, s_t)$.

Sous forme logarithmique, la formule précédente implique le score additif des sommets « prime pour les ions présents, pénalité pour les manquants » dans le graphe spectral (Dancik *et al.*, 1999 [79]).

11.6 Chemins anti-symétriques et séquençage peptidique

Après avoir construit le graphe spectral, on transforme le problème du séquençage peptidique en celui du *chemin le plus long dans un graphe acyclique orienté*, qui est résolu par un algorithme rapide de complexité temporelle linéaire. Malheureusement, cet algorithme simple ne fonctionne pas très bien dans la pratique. Le problème est que tout pic dans le spectre peut être interprété soit comme un ion N-terminal, soit comme un ion C-terminal. Par conséquent, chaque sommet « réel » (correspondant à une masse m) est associé à un sommet *jumeau* « virtuel » (correspondant à une masse $m(P) - m$). De plus, si le sommet réel a un score élevé, ce sera également le cas de son jumeau virtuel. Le plus long chemin dans le graphe spectral a donc tendance à inclure *à la fois* le sommet réel et son jumeau virtuel, car ils ont tous deux des scores élevés. De tels chemins ne correspondent pas à des reconstructions protéiques réalisables et devraient être évités. Cependant, les algorithmes connus pour trouver le plus long chemin ne le permettent pas.

Ainsi, cette réduction du problème du séquençage peptidique à celui du chemin le plus long est inadéquate. Nous allons maintenant formuler le problème du *plus long chemin anti-symétrique*, qui modélise la reconstruction de séquences peptidiques, de manière adéquate.

Soit G un graphe et soit T un ensemble de *paires interdites* de sommets de G (jumeaux). Un chemin dans G est dit anti-symétrique s'il contient au plus un sommet de chaque paire interdite. Le *problème du plus long chemin anti-symétrique* consiste à trouver un plus long chemin anti-symétrique dans G avec un ensemble T de paires interdites. Malheureusement, le problème du plus long chemin anti-symétrique est NP-complet (Garey et Johnson, 1979 [119]), ce qui indique qu'il y a peu de chances de trouver des algorithmes efficaces pour le résoudre. Cependant, ce résultat négatif n'implique pas qu'il n'y a aucun espoir de trouver un algorithme efficace pour le séquençage peptidique par spectrométrie de masse en tandem ; en effet, dans ce problème, les paires interdites ont une *structure spéciale.*

Dans un graphe spectral, les sommets sont modélisés par des nombres qui correspondent aux masses de peptides partiels potentiels. Deux paires interdites de sommets (x_1, y_1) et (x_2, y_2) sont dites *non chevauchantes* si les intervalles (x_1, y_1) et (x_2, y_2) ne se chevauchent pas. Un graphe G avec un ensemble de paires interdites est dit *propre* si deux paires interdites de sommets ne se chevauchent jamais. Le problème du séquençage peptidique par spectrométrie de masse en tandem correspond au problème du plus long chemin anti-symétrique dans un graphe propre. Dancik *et al.*, 1999 [79] ont proposé un algorithme efficace pour ce problème dans un graphe propre.

11.7 Le problème de l'identification peptidique

Pevzner *et al.*, 2000 [270] ont étudié le problème suivant :

Problème de l'identification peptidique Étant donnés une base de données de peptides, un spectre S, un ensemble d'ion-types Δ et un paramètre k, trouver un peptide qui présente l'appariement maximal avec le spectre S et qui est à moins de k mutations ou modifications d'une entrée de la base de données.

La difficulté majeure dans le problème de l'identification peptidique vient du fait que des peptides très similaires P_1 et P_2 peuvent avoir des spectres S_1 et S_2 très différents. Notre but est de définir une notion de similitude spectrale en correspondance avec la similitude séquentielle. En d'autres termes, si P_1 et P_2 sont distants de quelques substitutions, insertions, suppressions ou modifications, la similitude spectrale entre S_1 et S_2 doit être grande. Évidemment, le dénombrement des pics communs est une mesure intuitive de la similitude spectrale. Cependant, elle diminue très rapidement lorsque le nombre de mutations augmente, amenant ainsi des limites dans la détection de similitudes dans une recherche sur base de données MS/MS. De surcroît, il y a de nombreuses corrélations entre les spectres de peptides voisins et seule une petite portion d'entre elles est captée par le dénombrement de « pics communs ». L'algorithme PEDANTA (Pevzner *et al.*, 2000 [270]) capte *toutes* les corrélations entre les spectres voisins pour tout k ; il traite les cas dans lesquels des mutations dans le peptide changent sensiblement le modèle de fragmentation. Par exemple, remplacer des acides aminés comme H, K, R et P peut altérer considérablement la fragmentation. Même dans un cas extrême — comme lorsqu'une seule mutation fait passer du modèle de fragmentation « seulement des b-ions » à celui « seulement des y-ions » — PEDANTA révèle la similitude entre les spectres correspondants.

11.8 Circonvolution spectrale

Soient S_1 et S_2 deux spectres. On définit la *circonvolution spectrale* comme $S_2 \ominus S_1 = \{s_2 - s_1 : \; s_1 \in S_1, s_2 \in S_2\}$; soit $(S_2 \ominus S_1)(x)$ la multiplicité de l'élément x dans cet ensemble. En d'autres termes, $(S_2 \ominus S_1)(x)$ est le nombre de paires $(s_1, s_2) \in S_1 \times S_2$ qui vérifient $s_2 - s_1 = x$. Si $M(P)$ est la masse parente du peptide P avec le spectre S, alors $S^R = M(P) - S$ est le *spectre inverse* de S (chaque b-ion (resp. y-ion) dans S correspond à un y-ion (resp. b-ion) dans S^R). La *circonvolution spectrale inverse* $(S_2 \ominus S_1^R)(x)$ est le nombre de paires $(s_1, s_2) \in S_1 \times S_2$ qui vérifient $s_2 + s_1 - M(P) = x$.

Pour illustrer cette approche, on considère deux copies P_1 et P_2 du même peptide. Le nombre de pics communs à S_1 et S_2 (dénombrement des pics communs) est la valeur de $S_2 \ominus S_1$ en $x = 0$. De nombreux algorithmes de recherche sur base de données MS/MS tentent implicitement de trouver un peptide P

dans la base de données qui maximise $S_2 \ominus S_1$ en $x = 0$, où S_2 est un spectre expérimental et S_1 un spectre théorique du peptide P. Cependant, si l'on commence à introduire k mutations dans P_2 par rapport à P_1, la valeur de $S_2 \ominus S_1$ en $x = 0$ diminue rapidement. Par suite, le pouvoir de discernement du dénombrement des pics communs ($S_2 \ominus S_1$ en $x = 0$) chute sensiblement en $k = 1$ et disparaît presque pour $k > 1$.

Les pics dans la circonvolution spectrale permettent de détecter les mutations et les modifications sans recherche exhaustive. Supposons que P_2 ne diffère de P_1 que d'une mutation ($k = 1$), avec une différence d'acides aminés de $\delta = M(P_2) - M(P_1)$. Dans ce cas, on s'attend à ce que $S_2 \ominus S_1$ ait deux pics approximativement égaux en $x = 0$ et $x = \delta$. Si la mutation a lieu en position t dans le peptide, alors le pic en $x = 0$ correspond à des b_i-ions pour $i < t$ et à des y_i-ions pour $i \geq t$. Le pic en $x = \delta$ correspond à des b_i-ions pour $i \geq t$ et à des y_i-ions pour $i < t$. Une mutation dans P_2 qui change $M(P_1)$ de δ « mute » également le spectre S_2 en décalant des pics de δ. Par suite, le nombre de pics communs à S_1 et au S_2 « muté » peut augmenter en comparaison du nombre de pics communs à S_1 et S_2. Cette augmentation est bornée par $(S_2 \ominus S_1)(\delta)$ et $(S_2 \ominus S_1)(0) + (S_2 \ominus S_1)(\delta)$ est une borne supérieure pour le nombre de pics communs à S_1 et au S_2 « muté ».

L'autre ensemble des corrélations entre les spectres de peptides mutés est capté par la circonvolution spectrale inverse $S_2 \ominus S_1^R$, qui reflète les appariements d'ions N-terminaux et C-terminaux. On peut s'attendre à ce que $S_2 \ominus S_1^R$ ait deux pics aux *mêmes* positions 0 et δ.

Supposons désormais que P_2 et P_1 soient distants de deux substitutions, dont l'une a une différence de masse δ_1 et l'autre $\delta - \delta_1$. Ces mutations produisent deux nouveaux pics dans la circonvolution spectrale en $x = \delta_1$ et en $x = \delta - \delta_1$. Pour une distribution uniforme des mutations dans un peptide aléatoire, le taux des hauteurs moyennes des pics en $0, \delta, \delta_1, \delta - \delta_1$ est $2:2:1:1$.

Pour augmenter le taux signal-contre-parasite, on combine les pics dans la circonvolution spectrale et la circonvolution spectrale inverse :

$$S = S_2 \ominus S_1 \ + \ S_2 \ominus S_1^R.$$

En outre, on combine les pics en 0 et en δ (tout comme en δ_1 et en $\delta - \delta_1$) en introduisant la *fonction de décalage*

$$F(x) = \frac{1}{2}(S(x) + S(\delta - x)).$$

Notons que $F(x)$ est symétrique par rapport à la droite d'équation $x = \frac{\delta}{2}$, avec $F(0) = F(\delta)$ et $F(\delta_1) = F(\delta - \delta_1)$. On s'intéresse aux pics de $F(x)$ pour $x \geq \frac{\delta}{2}$.

On définit $x_1 = \delta = M(P_2) - M(P_1)$ et $y_1 = F(\delta) = F(0)$. Soient $y_2 = F(x_2), y_3 = F(x_3), \ldots, y_k = F(x_k)$ les $k - 1$ plus grands pics de $F(x)$ pour $x \geq \delta/2$ et $x \neq \delta$. On définit :

$$SIM_k(S_1, S_2) = \sum_{i=1}^{k} y_i$$

comme étant une estimation de la similitude entre les spectres S_1 et S_2 sous l'hypothèse que les peptides correspondants sont à distance k. Habituellement, SIM_k est la hauteur globale des k plus grands pics de la fonction de décalage. Par exemple, $SIM_1(S_1, S_2) = y_1$ est une borne supérieure pour le nombre de pics communs à S_1 et au S_2 « muté », si l'on autorise $k = 1$ mutation dans P_2.

Bien que la circonvolution spectrale aide à identifier les peptides mutés, elle est sérieusement limitée pour les raisons suivantes. Soit :

$$S = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$$

un spectre du peptide P ; pour simplifier, supposons que P ne produise que des b-ions. Soient :

$$S' = \{10, 20, 30, 40, 50, 55, 65, 75, 85, 95\}$$

et

$$S'' = \{10, 15, 30, 35, 50, 55, 70, 75, 90, 95\}$$

deux spectres théoriques correspondant aux peptides P' et P'' de la base de données. Quel peptide (P' ou P'') s'ajuste le mieux au spectre S ? Le dénombrement des pics communs ne permet pas de répondre à cette question car S' et S'' ont tous deux cinq pics en commun avec S. En outre, la circonvolution spectrale ne répond pas non plus à la question, car $S \ominus S'$ et $S \ominus S''$ (ainsi que les fonctions de décalage correspondantes) révèlent de forts pics de même hauteur en 0 et en 5. Ceci suggère que P' et P'' peuvent être obtenus à partir de P par une simple mutation de différence de masse égale à 5. Cependant, une analyse plus fine montre que, bien que cette mutation puisse être réalisée pour P' en introduisant un décalage de 5 après la masse 50, il ne peut être réalisé pour P''. La différence majeure entre S' et S'' est que les positions d'appariement dans S' viennent en groupe, ce qui n'est pas le cas pour celles de S''. Décrivons maintenant le principe de l'alignement spectral, qui aborde ce problème.

11.9 Alignement spectral

Soit $A = \{a_1, \ldots, a_n\}$ un ensemble ordonné de nombres entiers naturels, avec $a_1 < a_2 \ldots < a_n$. Un *décalage* Δ_i transforme A en $\{a_1, \ldots a_{i-1}, a_i + \Delta_i, \ldots, a_n + \Delta_i\}$. On ne considère que les décalages qui ne changent pas l'ordre des éléments, c'est-à-dire les décalages qui vérifient $\Delta_i \geq a_{i-1} - a_i$. Étant donnés des ensembles $A = \{a_1, \ldots, a_n\}$ et $B = \{b_1, \ldots, b_m\}$, on veut trouver une série de k décalages de A qui rendent A et B aussi similaires que possible. La *k-similitude* $D(k)$ entre les ensembles A et B est définie comme étant le nombre maximal d'éléments communs à ces ensembles après k décalages. Par exemple, un décalage -5_6 transforme :

$$S = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$$

en :

$$S' = \{10, 20, 30, 40, 50, 55, 65, 75, 85, 95\}$$

et, par conséquent, $D(1)$ vaut 10 pour ces ensembles. L'ensemble :

$$S'' = \{10, 15, 30, 35, 50, 55, 70, 75, 90, 95\}$$

a cinq éléments en commun avec S (les mêmes que S'), mais il n'existe pas de décalage transformant S en S'' et $D(1)$ vaut 6. Ci-dessous, on décrit un algorithme de programmation dynamique pour calculer $D(k)$.

On définit un *produit spectral* $A \otimes B$ comme étant une matrice de taille $a_n \times b_m$ contenant nm 1 (correspondant à toutes les paires d'indices (a_i, b_j)) et des zéros partout ailleurs. Le nombre de 1 sur la diagonale principale de cette matrice décrit le nombre de pics communs aux spectres A et B ou, en d'autres termes, la 0-similitude entre A et B. La figure 11.7 montre les produits spectraux $S \otimes S'$ et $S \otimes S''$ pour l'exemple provenant de la section précédente. Dans les deux cas, le nombre de 1 sur la diagonale principale est le même et $D(0)$ vaut 5. Le dénombrement des pics δ-décalés est égal au nombre de 1 sur la diagonale $(i, i + \delta)$. L'inconvénient de la fonction de décalage est qu'elle considère les diagonales séparément, sans combiner tous les scénarios de mutation possibles. La *k-similitude* entre les spectres est définie comme le nombre maximal de 1 sur un chemin qui passe à travers la matrice spectrale et qui utilise au plus $k + 1$ diagonales ; l'*alignement spectral k-optimal* est défini comme un chemin qui utilise ces $k + 1$ diagonales. Par exemple, la 1-similitude est définie comme le nombre maximal de 1 sur un chemin qui passe au travers de cette matrice et qui utilise au plus deux diagonales. La figure 11.7 révèle que la notion de 1-similitude permet de découvrir que S est plus proche de S' que de S'' car, dans le premier cas, le chemin 2-diagonal recouvre dix nombres 1 (matrice de gauche), contre six dans le second cas (matrice de droite). La figure 11.8 illustre le fait que l'alignement spectral permet de détecter de plus en plus de similitudes subtiles entre les spectres en augmentant k. Ci-dessous, on décrit un algorithme de programmation dynamique pour l'alignement spectral.

Soient A_i le i-préfixe de A et B_j le j-préfixe de B. On définit $D_{ij}(k)$ comme étant la k-similitude entre A_i et B_j, de sorte que les derniers éléments de A_i et B_j s'apparient. Autrement dit, $D_{ij}(k)$ est le nombre maximal de 1 sur un chemin vers (a_i, b_j) qui utilise au plus $k+1$ diagonales. On dit que (i', j') et (i, j) sont *co-diagonales* si elles vérifient $a_i - a_{i'} = b_j - b_{j'}$ et on écrit $(i', j') < (i, j)$ si l'on a $i' < i$ et $j' < j$. Pour ce qui est des conditions initiales, on introduit un élément fictif $(0, 0)$ avec $D_{0,0}(k) = 0$ et on suppose que $(0, 0)$ est co-diagonal avec tout autre (i, j). $D_{ij}(k)$ se calcule par récurrence selon la règle :

$$D_{ij}(k) = \max_{(i',j')<(i,j)} \begin{cases} D_{i'j'}(k) + 1, & \text{si } (i', j') \text{ et } (i, j) \text{ sont co-diagonales ;} \\ D_{i'j'}(k-1) + 1, & \text{sinon.} \end{cases}$$

La k-similitude entre A et B est donnée par $D(k) = \max_{ij} D_{ij}(k)$.

L'algorithme de programmation dynamique décrit pour l'alignement spectral est plutôt lent (la complexité temporelle est $O(n^4 k)$ pour des spectres de

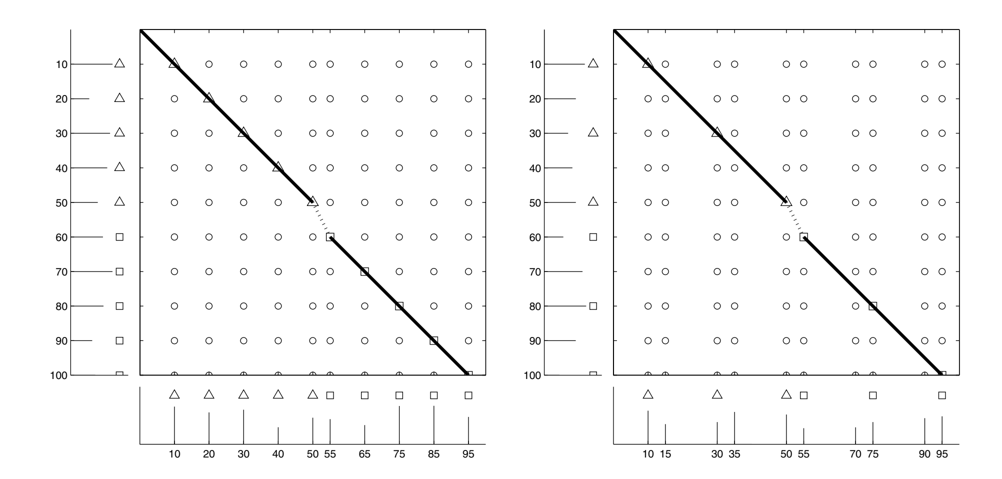


Figure 11.7 – Le spectre S peut être transformé en S' par une simple mutation et $D(1)$ vaut 10 (matrice de gauche). Le spectre S ne peut être transformé en S'' par une seule mutation et $D(1)$ est égal à 6 (matrice de droite).

n éléments). Voici un algorithme en $O(n^2k)$ pour résoudre le problème. On définit $diag(i,j)$ comme étant la paire co-diagonale maximale de (i,j) qui vérifie $diag(i,j) < (i,j)$. En d'autres termes, $diag(i,j)$ est la position du précédent 1 sur la même diagonale que (a_i, b_j) et, s'il n'en existe pas, il vaut $(0,0)$. On définit :

$$M_{ij}(k) = max_{(i',j') \leq (i,j)} D_{i'j'}(k).$$

Alors la récurrence pour $D_{ij}(k)$ peut être réécrite de la manière suivante :

$$D_{ij}(k) = \max \begin{cases} D_{diag(i,j)}(k) + 1, \\ M_{i-1,j-1}(k-1) + 1 \end{cases}$$

La récurrence pour $M_{ij}(k)$ est donnée par :

$$M_{ij}(k) = \max \begin{cases} D_{ij}(k) \\ M_{i-1,j}(k) \\ M_{i,j-1}(k) \end{cases}$$

La transformation décrite du graphe de programmation dynamique est obtenue en introduisant des arêtes horizontales et verticales qui fournissent le décalage entre les diagonales (figure 11.9). Le score d'un chemin est le nombre de 1 situés sur ce chemin, tandis que k correspond au nombre de décalages (nombre de diagonales utilisées moins 1).

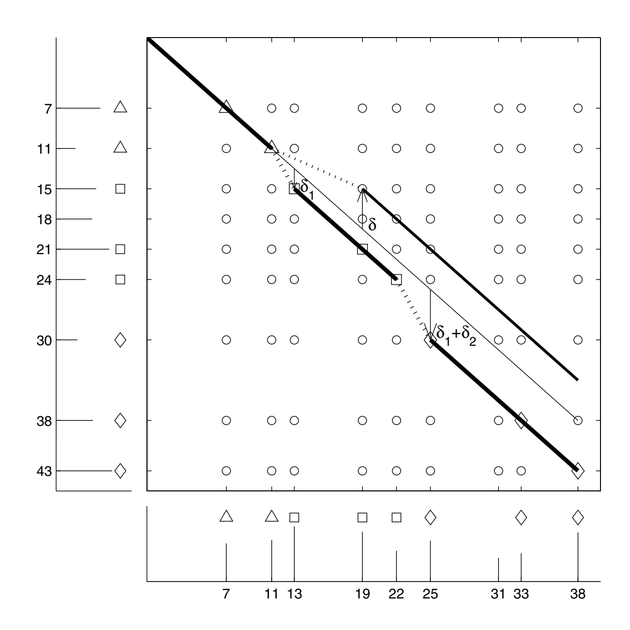


Figure 11.8 – Alignement de spectres. Le dénombrement de pics communs révèle seulement $D(0) = 3$ pics d'appariement sur la diagonale principale, tandis que l'alignement spectral révèle davantage de similitudes cachées entre les spectres ($D(1) = 5$ et $D(2) = 8$) et détecte les mutations correspondantes.

11.10 Alignement de peptides contre des spectres

La description simple précédente cache de nombreux détails qui rendent le problème de l'alignement spectral difficile. Un spectre est habituellement la combinaison d'une série de nombres croissante (les ions N-terminaux) et d'une série de nombres décroissante (les ions C-terminaux). Ces séries forment deux diagonales dans le produit spectral $S \otimes S$, la diagonale principale et sa perpendiculaire, qui correspond aux appariements d'ions N-terminaux et C-terminaux. L'algorithme décrit ne capte pas cet aspect et ne s'occupe que de la diagonale principale.

Pour combiner les séries N-terminales et C-terminales, on travaille avec $(S_1 \cup S_1^R) \otimes (S_2 \cup S_2^R)$, où S^R est le spectre inverse du peptide P. Cette transformation crée une « b-version » pour chaque y-ion et une « y-version » pour chaque b-ion, augmentant ainsi les parasites (car chaque pic parasite se propage deux fois). Une autre difficulté (plus sérieuse) est que chaque 1 dans le produit spectral aura un jumeau inverse et que seul l'un de ces jumeaux doit être comptabilisé dans l'alignement faisable. Ignorer ce problème peut mener à des solutions irréalisables qui sont éliminées dans l'approche du chemin antisymétrique (Dancik *et al.*, 1999 [79]).

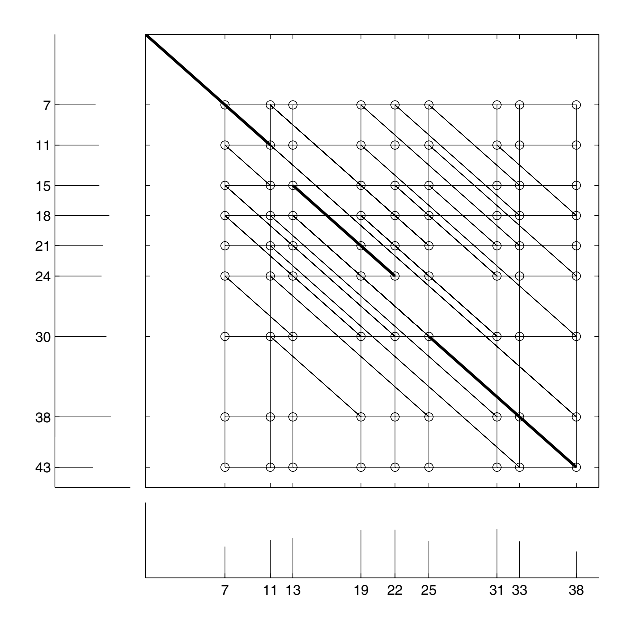


Figure 11.9 – La modification d'un graphe de programmation dynamique aboutit à un algorithme rapide d'alignement spectral.

L'algorithme décrit ne capte pas non plus tous les détails pertinents dans le cas de la comparaison « séquence contre spectre ». Dans ce cas, les arcs horizontaux et verticaux dans le graphe de programmation dynamique (figure 11.9) sont limités par les décalages possibles, qui reflètent les différences de masses entre les acides aminés participant à la mutation. Soit $P = p_1 \ldots p_n$ un peptide que l'on compare au spectre $S = \{s_1, \ldots, s_m\}$. Le d-préfixe du spectre S contient tous les pics de S avec $s_i \leq d$. On introduit une nouvelle variable $H_{i,d}(k)$, qui décrit la « meilleure » transformation du i-préfixe du peptide P en le d-préfixe du spectre S, avec au plus k substitutions dans P_i. Plus précisément, $H_{i,d}(k)$ décrit le nombre de 1 situés sur le chemin optimal avec k décalages entre les diagonales allant de $(0, 0)$ à la position (i, d) de la matrice $P \otimes S$ « peptide contre spectre » bien définie. Dans un souci de simplicité, on suppose que le spectre théorique de P ne contient que des b-ions.

Soit $H_{i,d}(k)$ la « meilleure » transformation de P_i en S_d avec k substitutions (autrement dit, une transformation qui utilise le nombre maximal de 1 sur un chemin avec au plus k décalages entre les diagonales). Cependant, dans ce cas, les sauts entre les diagonales ne sont pas arbitraires mais restreints par les différences de masses des acides aminés mutés (ou les différences de masses correspondant aux modifications chimiques). Ci-dessous, on décrit l'algorithme de programmation dynamique dans le cas de substitutions (les suppressions, les insertions et les modifications aboutissent à des récurrences semblables).

On définit $x(d) = 1$ si d est un élément de S et $x(d) = 0$ sinon. Alors $H_{i,d}(k)$ est décrit par la récurrence suivante ($m(a)$ est la masse de l'acide aminé a) :

$$H_{i,d}(k) = \max \begin{cases} H_{i-1,d-m(p_i)}(k) + x(d) \\ \max_{a=1,20} H_{i,d-(m(a)-m(p_i))}(k-1) \end{cases}$$

11.11 Quelques autres problèmes et approches

11.11.1 De la protéomique à la génomique

La spectrométrie de masse a beaucoup de succès pour l'identification de protéines dont les gènes sont contenus dans les bases de données de séquences. Cependant, l'interprétation *de novo* des spectres de masse en tandem est restée un problème complexe et coûteux en temps et, par suite, la spectrométrie de masse n'a pas encore eu d'impact significatif pour la découverte de *nouveaux* gènes. En 1995, Mann et Wilm (Mann et Wilm, 1995 [231]) ont remarqué qu'ils ne pouvaient trouver dans la littérature un seul exemple de gène cloné *essentiellement* sur la base d'une information séquentielle dérivée de MS/MS. Cette situation a changé ces cinq dernières années ; en particulier, les études de génétique inverse sur la sous-unité catalytique de la télomérase (Lingner *et al.*, 1997 [223]) ont nécessité le séquençage de novo de quatorze peptides avec la création ultérieure d'amorces PCR pour l'amplification génétique.

11.11.2 Analyse protéique à grande échelle

Les mélanges protéiques complexes peuvent être séparées par électrophorèse sur gel bidimensionnelle de haute résolution. Après la séparation, l'identité de chaque « zone » (peptide) dans le gel 2-D est inconnue et doit être identifiée par spectrométrie de masse. Cette approche nécessite des méthodes efficaces pour extraire des peptides obtenus à partir du gel et les transférer dans le spectromètre de masse.

Chapitre 12

Problèmes

12.1 Introduction

La biologie moléculaire a motivé de nombreux problèmes combinatoires intéressants. Il y a quelques années, Pevzner et Waterman, 1995 [275] ont compilé une collection de 57 problèmes ouverts en bio-informatique moléculaire. Cinq ans après, un quart d'entre eux avaient été résolus. C'est pour cette raison que je ne précise pas explicitement si les problèmes listés ci-dessous sont encore ouverts : certains auront peut-être été résolus au moment où le lecteur consultera cet ouvrage.

12.2 Cartographie de restriction

Supposons qu'une molécule d'ADN soit digérée deux fois, par deux enzymes de restriction. Le graphe d'intervalles des fragments obtenus est un *graphe d'intervalles biparti* (Waterman et Griggs, 1986 [361]).

Problème 12.1 *Concevoir un algorithme efficace reconnaissant un graphe d'intervalles biparti.*

Problème 12.2 *Soit v un sommet de degré pair dans un graphe colorié équilibré. Prouver que les $d(v)$ arêtes incidentes à v peuvent être partitionnées en $d(v)/2$ paires, de sorte que les arêtes d'une même paire ne soient pas de la même couleur.*

Soit $G(V, E)$ un graphe l-colorié équilibré et soit $P = x_1 \dots x_m$ un chemin dans G, d'ensemble d'arêtes EP. Une couleur c est dite *critique* pour P si (i) elle est différente de la couleur de la dernière arête (x_{m-1}, x_m) de P et si (ii) c'est la couleur la plus fréquente parmi les arêtes de $E \setminus EP$ incidentes à x_m. Les arêtes de l'ensemble $E \setminus EP$ qui sont incidentes à x_m et qui ont une couleur

critique sont appelées des arêtes *critiques*. Un chemin est dit critique s'il est obtenu en choisissant une arête critique à chaque étape.

Problème 12.3 *Montrer comment utiliser les chemins critiques pour construire des cycles eulériens alternés.*

Le problème suivant demande un résultat analogue au théorème BEST pour les graphes bicolores.

Problème 12.4 *Trouver le nombre de cycles eulériens alternés dans un graphe eulérien bicolore.*

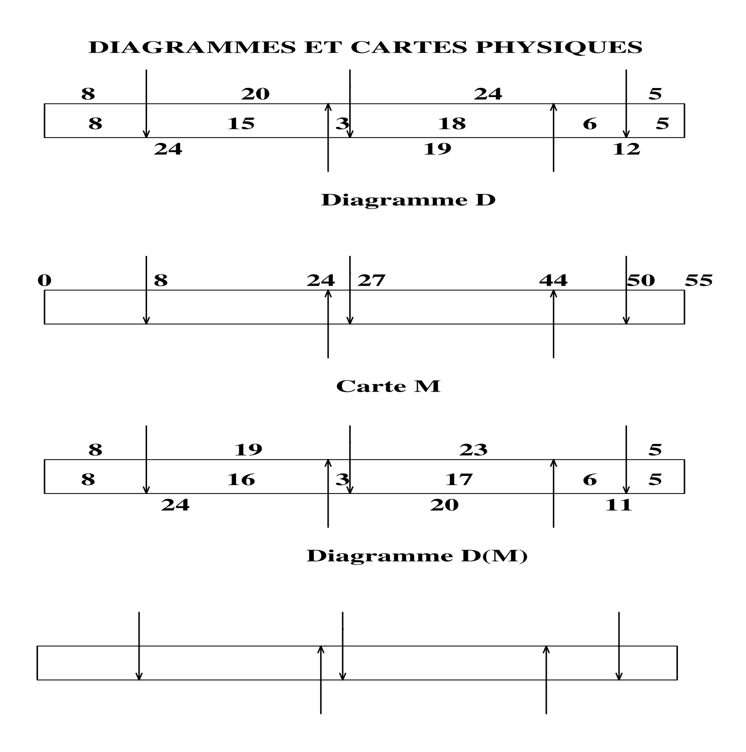


Figure 12.1 – Diagrammes et cartes physiques.

La plupart des algorithmes pour le problème de la double digestion sont fondés sur la production et l'analyse d'hypothèses concernant l'ordre des fragments de restriction dans une carte physique. Chacune de ces hypothèses correspond à un *diagramme de cartographie* D qui exhibe l'ordre des sites et la longueur des fragments (figure 12.1). Notons que ce diagramme ne montre pas les coordonnées des sites. Une carte physique M fournit de l'information sur l'ordre et sur les coordonnées des sites. Toute carte physique correspond à un diagramme $D(M)$ (figure 12.1), où les longueurs des fragments correspondent

à la distance entre les sites. L'inverse n'est pas vrai : un diagramme ne correspond pas nécessairement à une carte physique. On peut alors se demander comment construire une carte physique la plus proche possible du diagramme. Soient $D = (d_1, \ldots, d_n)$ et $M = (m_1, \ldots, m_n)$ les longueurs de tous les fragments dans les diagrammes D et $D(M)$, dans cet ordre. La distance entre le diagramme D et la carte M est :

$$d(D, M) = \max_{i=1,n} |d_i - m_i|.$$

Par exemple, dans la figure 12.1, on a $D = (8, 20, 24, 5, 24, 19, 12, 8, 15, 3, 18, 6, 5)$, $M = (8, 19, 23, 5, 24, 20, 11, 8, 16, 3, 17, 6, 5)$ et $d(D, M) = 1$. Le *problème de l'ajustement de diagrammes* consiste à trouver une carte située le plus près possible d'un diagramme :

Problème 12.5 *Étant donné un diagramme D, trouver une carte M qui minimise $d(D, M)$.*

Problème 12.6 *Étant données deux cartes dans la même classe d'équivalence, trouver l'une des plus courtes séries de transformations de cassettes faisant passer d'une carte à l'autre.*

Voici une généralisation du problème de la double digestion : on a trois enzymes A, B et C et on obtient des données expérimentales sur les longueurs des fragments lors des digestions *simples* A, B et C, des digestions *doubles* AB, BC et CA et de la digestion *triple* ABC. Une telle expérimentation mène au *problème de la digestion multiple.*

Problème 12.7 *Trouver une carte physique de trois enzymes A, B et C, sachant que l'on fournit six ensembles de données expérimentales (les digestions A, B, C, AB, BC, CA et ABC).*

Problème 12.8 *Caractériser les transformations de cassettes de cartes multiples (trois enzymes ou plus).*

L'algorithme PDP de Rosenblatt et Seymour, 1982 [289] est pseudo-polynomial.

Problème 12.9 *Existe-t-il un algorithme polynomial pour le PDP ?*

Skiena *et al.*, 1990 [314] ont prouvé que $H(n)$, le nombre maximal d'ensembles fortement homométriques sur n éléments vérifie $\frac{1}{2}n^{0,6309} \leq H(n) \leq \frac{1}{2}n^{2,5}$. La borne supérieure semble plutôt pessimiste.

Problème 12.10 *Trouver des bornes mieux ajustées pour $H(n)$.*

Problème 12.11 *Prouver que tout ensemble de cinq points est reconstructible.*

Problème 12.12 *Imaginer un algorithme efficace pour le problème de la digestion partielle sondée.*

Problème 12.13 *Trouver des bornes inférieure et supérieure pour le nombre maximal de solutions d'un problème de digestion partielle sondée avec n sites.*

Dans le problème de cartographie optique, la donnée initiale est une matrice $S = (s_{ij})$ de taille $n \times m$ formée de 0 et de 1 ; chaque ligne correspond à une molécule d'ADN (directe ou inversée), chaque colonne correspond à une position dans cette molécule et s_{ij} est égal à 1 s'il y a une coupure à la position j de la molécule i. Le but est de renverser l'orientation d'un sous-ensemble de molécules (sous-ensemble de lignes dans S) et de déclarer un sous-ensemble des t colonnes comme étant des « vrais sites de coupures », de sorte que le nombre de 1 soit maximisé dans les colonnes des sites de coupure (Karp et Shamir, 1998 [190]).

Une approche naïve pour ce problème consiste à trouver t colonnes ayant une grande proportion de 1 et à les déclarer comme étant des sites de coupure potentiels. Cependant, dans cette approche, chaque vrai site aura un jumeau inverse. Soit $w(i, j)$ le nombre de molécules où sont présents les deux sites de coupure i et j (soit dans l'orientation directe, soit en sens inverse). Dans une approche différente, on construit un graphe d'ensemble de sommets $\{1, \ldots, m\}$, dans lequel deux sommets sont reliés par une arête (i, j) de poids $w(i, j)$.

Problème 12.14 *Établir une relation entre la cartographie optique et le problème du plus long chemin anti-symétrique.*

12.3 Assemblage de cartes

Problème 12.15 *Trouver le nombre de chevauchements différents de n clones.*

Un chevauchement de n clones peut être déterminé par une suite d'entiers $a_1 \ldots a_n$, où a_i est le nombre de clones qui se terminent avant que le clone i ne commence. Par exemple, le chevauchement des neuf clones dans la figure 3.3 correspond à la suite 000112267. Toute suite d'entiers $a_1 \ldots a_n$ ne détermine pas un chevauchement valide. En outre, si une sonde détermine un intervalle de clones $[i, j]$, il implique les inégalités $a_j \leq i-1$ (les clones j et i se chevauchent) et $a_{j+1} \geq i - 1$ (les clones $i - 1$ et $j + 1$ ne se chevauchent pas).

Problème 12.16 *Formuler le problème de la plus courte chaîne couvrante avec un ordre de clones donné sous la forme d'un programme linéaire entier, puis le résoudre.*

Problème 12.17 *Soit $\mathcal{I}$ une collection d'intervalles sur une droite et soit k le nombre maximal d'intervalles deux à deux disjoints dans $\mathcal{I}$. Prouver qu'il existe k points sur la droite tels que chaque intervalle dans $\mathcal{I}$ contienne au moins l'un de ces points.*

Les graphes d'intersection correspondant aux collections d'arcs d'un cercle sont appelés des *graphes d'arcs circulaires*. Si une collection d'arcs d'un cercle ne couvre pas un point x de celui-ci, son *graphe d'arcs circulaires* est un graphe d'intervalles (il suffit de couper le cercle en x et de le redresser).

Problème 12.18 *Imaginer un algorithme efficace pour reconnaître les graphes d'arcs circulaires.*

Problème 12.19 *Si l'on choisit N clones aléatoires de longueur L provenant d'un génome de longueur G, la fraction attendue du génome représenté dans ces clones est approximativement $1 - e^c$, où $c = \frac{NL}{G}$ est la couverture.*

Dans la *cartographie de contigs de cosmides* (Zhang *et al.*, 1994 [376]), l'information sur le chevauchement des clones est fournie par les données d'hybridation. Un ensemble de clones est placé sur un filtre pour l'hybridation de colonies et le filtre est sondé avec un clone radioactif. Ce processus produit une information sur le chevauchement (quelles sondes chevauchent d'autres clones, par exemple). Si l'on n'utilise qu'un sous-ensemble de clones comme sondes, l'information sur le chevauchement n'est pas valable entre les clones qui ne sont pas des sondes. Un graphe est un *graphe d'intervalles de sondes* si son ensemble de sommets peut être partitionné en deux sous-ensembles P (les clones utilisés comme sondes) et N (les clones non utilisés commes sondes), avec un intervalle associé à chaque sommet, de sorte que deux sommets soient adjacents si et seulement si les intervalles correspondants se chevauchent et au moins un des sommets est dans P (McMorris *et al.*, 1998 [235]).

Problème 12.20 *Écrire un algorithme qui reconnaît les graphes d'intervalles de sondes.*

La *cartographie du produit interne* (Perlin et Chakravarti, 1993 [263]) est une approche de la cartographie de clones qui sonde deux fois un ensemble de clones d'hybrides de radiation (une fois avec les BAC et une fois avec les STS), afin d'obtenir une carte des BAC relative aux STS. La cartographie du produit interne nécessite deux ensembles de données : une matrice de criblage hybride avec des STS et une matrice de criblage hybride avec des BAC.

Problème 12.21 *Étant données des matrices de criblage hybrides avec des STS et des BAC, construire une carte des BAC relative aux STS.*

Les éléments $\pi_i\pi_j\pi_k$, pour $1 \leq i < j < k \leq n$, forment un *triplet ordonné* dans une permutation $\pi = \pi_1 \dots \pi_n$. Soit $\Phi(\pi)$ l'ensemble de tous les C_n^3 triplets ordonnés pour π. La cartographie par hybrides de radiation motive le problème suivant :

Problème 12.22 *Étant donné un ensemble arbitraire T de triplets ordonnés d'un ensemble à n éléments, trouver une permutation π qui vérifie $T \subset \Phi(\pi)$.*

Les éléments $\pi_i\pi_j\pi_k$ forment un *triplet non ordonné* s'ils vérifient soit $1 \leq j < k < i \leq n$, soit $1 \leq k < j < i \leq n$. Soit $\Theta(\pi)$ l'ensemble de tous les triplets non ordonnés pour π.

Problème 12.23 *Étant donné un ensemble arbitraire T de triplets non ordonnés d'un ensemble à n éléments, trouver une permutation π qui vérifie $T \subset \Theta(\pi)$.*

12.4 Séquençage

L'heuristique la plus simple pour le problème de la super-chaîne la plus courte est l'algorithme GLOUTON : on fusionne de façon répétée une paire de chaînes de chevauchement maximal, jusqu'à ce qu'il ne reste plus qu'une seule chaîne. Tarhio et Ukkonen, 1988 [333] ont défini la *compression* d'un algorithme SSP comme étant le nombre de symboles économisés par cet algorithme en comparaison de la simple concaténation de toutes les chaînes.

Problème 12.24 *Prouver que l'algorithme GLOUTON accomplit au moins la moitié de la compression d'une super-chaîne optimale ; autrement dit, il vérifie :* $\frac{\textit{compression GLOUTONNE}}{\textit{compression optimale}} \geq \frac{1}{2}$.

Une garantie de performance concernant la compression n'implique pas une garantie de performance sur la longueur. Comme on ne connaît pas d'exemple pour lequel le taux d'approximation de GLOUTON est pire que 2, Blum *et al.*, 1994 [37] ont émis la conjecture suivante :

Problème 12.25 *Prouver que GLOUTON garantit une performance de 2.*

Soient $\mathcal{S} = \{s_1, \ldots, s_n\}$ une collection de chaînes linéaires et $\mathcal{C} = \{c_1, \ldots, c_m\}$ une collection de chaînes circulaires. On dit que $\mathcal{C}$ est une *circulation* de $\mathcal{S}$ si chaque s_i est contenue dans l'une des chaînes circulaires c_j, pour $1 \leq j \leq m$. La longueur de circulation $|\mathcal{C}| = \sum_{j=1}^{m} |c_j|$ est la longueur totale des chaînes de $\mathcal{C}$.

Problème 12.26 *Trouver la plus courte circulation pour une collection de chaînes linéaires.*

Soient $P = \{s_1, \ldots, s_m\}$ un ensemble de chaînes *positives* et $N = \{t_1, \ldots, t_k\}$ un ensemble de chaînes *négatives*. On suppose qu'aucune chaîne t_i n'est une sous-chaîne d'une chaîne positive s_j. Une *super-chaîne compatible* pour (P, N) est une chaîne s telle que chaque s_i soit une sous-chaîne de s et qu'aucune t_i ne soit une sous-chaîne de s (Jiang et Li, 1994 [180]).

Problème 12.27 *Imaginer un algorithme d'approximation pour le problème de la plus courte super-chaîne compatible.*

Les courts fragments lus par séquençage contiennent des erreurs qui amènent des complications dans l'assemblage de fragments. L'introduction d'erreurs aboutit au *problème de la plus courte super-chaîne k-approximative* (Jiang et Li, 1996 [181]) :

Problème 12.28 *Étant donné un ensemble S de chaînes, trouver l'une des plus courtes chaînes w, telles que chaque chaîne x dans S s'apparie à une sous-chaîne de w avec au plus k erreurs.*

Supposons que l'on nous donne un ensemble $\mathcal{S}$ de n chaînes *aléatoires* de taille fixée l dans un alphabet de A lettres. Si n est grand (de l'ordre de A^l), la longueur de la plus courte super-chaîne commune $E(\mathcal{S})$ pour l'ensemble $\mathcal{S}$ est de l'ordre de n. Si n est petit, $E(\mathcal{S})$ est de l'ordre de nl.

Problème 12.29 *Estimer $E(\mathcal{S})$ en tant que fonction de l, n et A.*

Étant données des chaînes s et t, *chevauche*(s,t) est la longueur d'un préfixe maximal de s qui s'apparie à un suffixe de t.

Problème 12.30 *Étant donnée une collection de chaînes i.i.d. $\{s_1, \ldots, s_n\}$ de longueur fixée, trouver la distribution de* $\max\{\textit{chevauche}(s_i, s_j) : 1 \leq i \neq j \leq n\}$.

Étant donnés une collection de *reads* $\mathcal{S} = \{s_1, \ldots, s_n\}$ *provenant d'un projet de séquençage d'ADN et un entier l, le spectre de $\mathcal{S}$ est un ensemble $\mathcal{S}_l$ formé de tous les l-uplets des chaînes $s_1, \ldots, s_n$. Soit Δ une borne supérieure pour le nombre d'erreurs dans chaque* read d'ADN. Une approche possible pour le problème de l'assemblage de fragments est de commencer par corriger les erreurs dans chaque *read, puis d'assembler les* reads corrects en contigs. Ceci motive le problème suivant :

Problème 12.31 *Étant donnés $\mathcal{S}$, Δ et l, introduire jusqu'à Δ corrections dans chaque chaîne de $\mathcal{S}$, de sorte que $|\mathcal{S}_l|$ soit minimisé.*

12.5 Puces à ADN

Problème 12.32 *Prouver que la borne inférieure théorique pour le nombre de sondes nécessaires à la reconstruction sans ambiguïté d'une chaîne arbitraire de longueur n est $\Omega(n)$.*

Problème 12.33 *Imaginer un algorithme pour la reconstruction de séquences par SBH à partir de données comportant des erreurs (faux positif ou faux négatif).*

Étant données deux chaînes avec la même composition en l-uplets, la *distance* qui les sépare est la longueur de *la plus courte* série de transpositions transformant l'une en l'autre.

Problème 12.34 *Proposer un algorithme permettant de trouver la valeur exacte ou une approximation de la distance séparant deux chaînes avec la même composition en l-uplets.*

Problème 12.35 *Quelle est la plus grande distance entre deux chaînes de n lettres ayant la même composition en l-uplets ?*

L'*hybridation superposable continue* suppose une hybridation supplémentaire de courtes sondes qui étend continûment les duplex formés par le fragment d'ADN cible et les sondes de la puce de séquençage. Dans cette approche, l'hybridation supplémentaire avec un m-uplet court sur la puce $C(k)$ fournit de l'information sur des $(k+m)$-uplets contenus dans la séquence.

Problème 12.36 *Étant donné un spectre S ne fournissant pas de reconstruction par SBH non ambiguë, déterminer le nombre minimal d'expériences d'hybridation superposable continue nécessaires pour reconstruire le fragment cible de façon non ambiguë.*

Problème 12.37 *Reconstruire la séquence d'un fragment d'ADN, étant donnés un spectre S et les résultats d'hybridation superposable continue supplémentaires.*

Une *puce binaire réduite* d'ordre l est une puce de mémoire 2×2^l, composée de toutes les multisondes des deux sortes :

$$\underbrace{\{W,S\},\{W,S\},\ldots,\{W,S\}}_{l} \text{ et } \underbrace{\{R,Y\},\{R,Y\},\ldots,\{R,Y\}}_{l}.$$

Par exemple, pour $l = 2$, la puce binaire réduite est constituée de huit multisondes : $WW, WS, SW, SS, RR, RY, YR$ et YY. Chaque multisonde est un regroupement de quatre dinucléotides.

Problème 12.38 *Calculer la probabilité de ramification des puces binaires réduites et la comparer à celle des puces uniformes.*

On dit d'une puce qu'elle est *k-bornée* si toutes ses sondes sont de longueur inférieure ou égale à k.

Problème 12.39 *Étant donnés deux entiers m et k, trouver une puce k-bornée avec m (multi)sondes qui maximise la puissance de résolution.*

Voici une version plus facile du problème précédent :

Problème 12.40 *Les puces binaires $C_{bin}(k-1)$ fournissent-elles asymptotiquement la meilleure puissance de résolution parmi toutes les puces k-bornées ?*

Bien que les puces binaires fournissent une meilleure puissance de résolution que les puces uniformes, on ne connaît pas encore d'algorithme efficace pour la reconstruction d'un fragment d'ADN à partir de son spectre sur une puce binaire.

Problème 12.41 *Existe-t-il un algorithme polynomial pour la reconstruction de séquences par SBH par des puces binaires ?*

La preuve du théorème 5.6 considère les changements s_{i+1} en x, y et s_i en z, u. Elle suppose implicitement que les ensembles de sommets $\{x, y\}$ et $\{z, u\}$ ne se chevauchent pas.

Problème 12.42 *Adapter la preuve au cas où ces ensembles se chevauchent.*

Soit $\mathcal{L}$ un ensemble de chaînes. On considère un ensemble E (*données sur les chaînes prioritaires*) formé de toutes les paires ordonnées de l-uplets différents tels qu'ils apparaissent dans une chaîne de $\mathcal{L}$, dans l'ordre donné mais à des distances arbitraires. Chetverin et Kramer, 1993 [66] ont suggéré l'approche par *hybridation de brins nichés* pour les puces à ADN, qui aboutit au problème suivant (Rubinov et Gelfand, 1995 [291]) :

Problème 12.43 *Reconstruire $\mathcal{L}$ à partir des données sur les chaînes prioritaires.*

Les codes de Gray bidimensionnels sont optimaux pour minimiser la longueur du bord des puces à ADN uniformes. Cependant, pour une puce *arbitraire*, le problème consistant à minimiser la longueur globale des bords des masques photolithographiques n'a toujours pas été résolu.

Problème 12.44 *Trouver un arrangement de sondes dans une puce (arbitraire) qui minimise la longueur totale des bords des masques pour la production de puces photolithographiques.*

12.6 Comparaison de séquences

Ajuster une séquence V à une séquence W est un problème qui consiste à trouver une sous-chaîne W' de W qui maximise le score d'alignement $s(V, W')$, parmi toutes les sous-chaînes de W.

Problème 12.45 *Imaginer un algorithme efficace pour le problème d'ajustement.*

Problème 12.46 *Estimer le nombre d'alignements différents entre deux séquences à n lettres.*

Problème 12.47 *Concevoir un algorithme qui calcule le nombre d'alignements optimaux distincts entre deux chaînes.*

Problème 12.48 *Pour deux chaînes $v_1 \ldots v_n$ et $w_1 \ldots w_m$, montrer comment calculer, pour chaque (i, j), la valeur du meilleur alignement qui ajuste le caractère v_i avec le caractère w_j.*

Problème 12.49 *Pour un paramètre k, calculer l'alignement global entre deux chaînes, si l'on impose qu'il contienne au plus k trous (blocs d'indels consécutifs).*

Le problème de l'*alignement à k différences* consiste à trouver le meilleur alignement global entre deux chaînes V et W avec au plus k mésappariements, insertions ou délétions.

Problème 12.50 *Imaginer un algorithme d'alignement global à k différences qui compare des chaînes de n lettres en un temps $O(kn)$.*

Chao *et al.*, 1992 [64] ont décrit un algorithme qui aligne deux séquences dans une bande diagonale ; il nécessite seulement un temps de calcul $O(nw)$ et présente une complexité spatiale $O(n)$, où n désigne la longueur des séquences et w la largeur de la bande.

Problème 12.51 *Un alignement dans une bande diagonale peut-il être implémenté avec une complexité spatiale $O(w)$?*

Myers et Miller, 1988 [246] ont étudié le problème suivant :

Problème 12.52 *Développer une version linéaire (du point de vue spatial) de l'alignement de séquences global avec des pénalités de brèches affines.*

Huang et Miller, 1991 [169] ont étudié le problème suivant :

Problème 12.53 *Développer une version linéaire (du point de vue spatial) de l'algorithme d'alignement local.*

Dans l'approche spatialement efficace de l'alignement de séquences, le problème original de taille $n \times m$ se réduit à deux sous-problèmes de tailles $i \times \frac{m}{2}$ et $(n - i) \times \frac{m}{2}$. Dans une implémentation parallèle rapide de l'alignement de séquences, il est souhaitable d'avoir un *partitionnement équilibré*, qui sépare le problème initial en deux sous-problèmes de taille égale.

Problème 12.54 *Imaginer un algorithme d'alignement efficace au niveau spatial avec un partitionnement équilibré.*

Le score d'un alignement local n'est pas normalisé en fonction de la longueur de la région d'appariement. Par suite, on choisira plutôt un alignement local de score 100 et de longueur 100 qu'un alignement local de score 99 et de longueur 10, bien que celui-ci soit probablement plus important d'un point de vue biologique. Pour refléter la longueur de l'alignement local, le score d'alignement local $s(I, J)$ entre les sous-chaînes I et J peut être ajusté en divisant $s(I, J)$ par la longueur totale des régions alignées : $\frac{s(I,J)}{|I|+|J|}$. Le problème de l'*alignement local normalisé* consiste à trouver des sous-chaînes I et J qui maximisent $\frac{s(I,J)}{|I|+|J|}$, parmi toutes les sous-chaînes I et J avec $|I| + |J| \geq k$, où k est un seuil pour la longueur globale minimale de I et J.

Problème 12.55 *Imaginer un algorithme qui résolve le problème de l'alignement local normalisé.*

On dit qu'une chaîne X est une *superséquence* d'une chaîne V si V est une sous-séquence de X.

Problème 12.56 *Étant données des chaînes V et W, concevoir un algorithme qui trouve le plus courte superséquence pour V et W.*

Soit P un modèle de longueur n et soit T un texte de longueur m. Le problème de la *répétition en tandem* consiste à trouver un intervalle dans T qui présente le meilleur alignement global avec l'une des répétitions en tandem de P. Soit P^m la concaténation de P avec lui-même m fois. Le problème de la répétition en tandem revient à calculer l'alignement local entre P^m et T et l'algorithme d'alignement local standard résout ce problème en un temps $O(nm^2)$.

Problème 12.57 *Trouver une approche qui résolve le problème de la répétition en tandem en un temps $O(nm)$.*

Par définition, un alignement de chaînes circulaires est un alignement de chaînes linéaires formé en coupant (linéarisant) ces chaînes circulaires en des positions arbitraires.

Problème 12.58 *Trouver un alignement optimal (local et global) de chaînes circulaires.*

Un alignement local entre deux chaînes différentes A et B trouve une paire de sous-chaînes de similitude maximale (l'une dans A et l'autre dans B). Supposons que l'on veuille trouver une paire de sous-chaînes (non chevauchantes) de similitude maximale *à l'intérieur* de A (*problème de la répétition inexacte optimale*). Calculer l'alignement local entre A et lui-même ne résout pas le problème, car l'alignement obtenu peut correspondre à des sous-chaînes chevauchantes. Ce problème a été étudié par Miller (manuscrit non publié), puis par Kannan et Myers, 1996 [184] et Schmidt, 1998 [308].

Problème 12.59 *Imaginer un algorithme pour le problème de la répétition inexacte optimale.*

Schoniger et Waterman, 1992 [310] ont étendu la portée des opérations d'édition dans l'alignement de séquences pour inclure les inversions *non chevauchantes* en plus des insertions, suppressions et substitutions.

Problème 12.60 *Trouver un algorithme efficace pour l'alignement de séquences avec des inversions non chevauchantes.*

Dans le problème de l'*alignement chimérique* (Komatsoulis et Waterman, 1997 [205]), on donne une chaîne V et une base de données de chaînes $\mathcal{W} = \{W_1, \ldots W_N\}$ et le problème consiste à trouver $\max_{1 \leq i \neq j \leq N} s(V, W_i \oplus W_j)$, où $W_i \oplus W_j$ est la concaténation de W_i et W_j.

Problème 12.61 *Imaginer un algorithme efficace pour le problème de l'alignement chimérique.*

Problème 12.62 *Montrer que, dans toute permutation de n entiers distincts, il y a soit une sous-séquence croissante de longueur supérieure ou égale à $\sqrt{n}$, soit une sous-séquence décroissante de longueur supérieure ou égale à $\sqrt{n}$.*

Une suite de *Catalan* est une permutation $x_1 \ldots x_{2n}$ de n 1 et n 0 telle que, dans tout préfixe $x_1 \ldots x_i$, le nombre de 1 soit supérieur ou égal au nombre de 0. Le nombre de telles suites est appelé le n-ième nombre de *Catalan* C_n.

Problème 12.63 *Prouver les assertions suivantes :*

- C_n *est le nombre de tableaux de Young standards avec deux lignes de longueur* n.
- C_n *est le nombre de permutations* $\pi \in S_n$ *ayant une plus longue sous-séquence décroissante de longueur au plus* 2.
- C_n *est le nombre de suites d'entiers positifs* $1 \leq a_1 \leq a_2 \leq \ldots \leq a_n$ *qui vérifient* $a_i \leq i$ *pour tout* i.

Problème 12.64 *Prouver la récurrence* $C_{n+1} = C_n C_o + C_{n-1} C_1 + \ldots + C_o C_n$.

Problème 12.65 *Prouver que la longueur de la plus longue sous-séquence décroissante d'une permutation π est égale à la longueur de la première colonne du tableau de Young $P(\pi)$.*

Une sous-séquence σ d'une permutation π est dite *k-croissante* si, en tant qu'ensemble, elle peut s'écrire sous la forme

$$\sigma = \sigma_1 \cup \sigma_2 \cup \ldots \cup \sigma_k,$$

où chaque σ_i est une sous-séquence croissante de π.

Problème 12.66 *Imaginer un algorithme qui permette de trouver les plus longues sous-séquences k-croissantes.*

Chang et Lampe [61] ont proposé une conjecture analogue à celle de Sankoff-Mainville pour la distance d'édition $d(V, W)$ entre des chaînes V et W i.i.d. à n lettres :

Problème 12.67 *Pour des chaînes aléatoires i.i.d. à n lettres dans un alphabet à k lettres, on a :*

$$\frac{Espérance(d(V,W))}{n} = 1 - \frac{1}{\sqrt{k}} + o(\frac{1}{\sqrt{k}}).$$

Gusfield *et al.*, 1994 [146] ont prouvé que le nombre de polygones convexes dans la décomposition de l'espace paramétrique pour l'alignement *global* était en $O(n^{2/3})$, où n est la longueur des séquences. Fernandez-Baca *et al.*, 1999 [102] ont étudié le problème suivant :

Problème 12.68 *Produire une paire de séquences de longueur n qui possèdent de l'ordre de $\Omega(n^{2/3})$ régions dans la décomposition de l'espace paramétrique.*

Pour un alphabet de longueur fixée, on ne connaît pas d'exemple de séquence avec $\Omega(n^{2/3})$ régions dans l'espace paramétrique.

Problème 12.69 *Améliorer la borne $O(n^{2/3})$ pour le nombre de régions dans la décomposition spatiale pour l'alignement global, dans le cas d'un alphabet borné.*

Problème 12.70 *Trouver des bornes pour l'espérance du nombre de régions dans la décomposition spatiale pour l'alignement global de deux séquences aléatoires de longueur n.*

Problème 12.71 *Généraliser les bornes pour le nombre de régions dans la décomposition spatiale, dans le cas de l'alignement multiple.*

La décomposition de l'espace paramétrique pour l'alignement local contient habituellement plus de régions que celle pour l'alignement global. Vingron et Waterman, 1994 [346] ont étudié les liens existant entre l'alignement de séquences paramétrique et la transition de phase. Dans cette relation, il est intéressant d'étudier la décomposition paramétrique du secteur logarithmique.

Problème 12.72 *Trouver des bornes pour l'espérance du nombre de régions dans la décomposition spatiale du secteur logarithmique pour l'alignement local de deux séquences aléatoires de longueur n.*

L'algorithme de Gusfield *et al.*, 1994 [146] pour l'alignement paramétrique de deux séquences fonctionne en un temps $O(R + E)$ par région, où R est le nombre de régions dans la décomposition paramétrique et E le temps nécessaire pour réaliser un simple alignement. Dans le cas de systèmes de scores non pondérés, on a $R = O(E)$; le coût par région est donc $O(E)$. Lorsque l'on utilise une matrice de poids, on ne sait pas grand'chose sur R. Gusfield a formulé le problème suivant :

Problème 12.73 *Estimer le nombre de régions dans une décomposition convexe, dans le cas de matrices de poids.*

Problème 12.74 *Imaginer un algorithme rapide pour la décomposition spatiale, dans le cas de matrices de poids.*

Comme les paramètres *énergétiques* pour le pliage de l'ARN sont estimés avec des erreurs, il serait utile d'étudier le pliage paramétrique de l'ARN. Par exemple, la comparaison de régions correspondant à des croisements pour la décomposition de l'espace paramétrique de l'ARNt fournirait une estimation de la précision des paramètres énergétiques de l'ARN utilisés habituellement.

Problème 12.75 *Développer un algorithme pour le pliage paramétrique de l'ARN.*

Soit $S_n(\mu, \delta)$ une variable aléatoire correspondant au score (# appariements – μ# mésappariements – σ# indels) de l'alignement global entre deux chaînes aléatoires i.i.d. de longueur n. Arratia et Waterman, 1994 [14] ont défini $a(\mu, \delta) = \lim_{n\to\infty} \frac{S_n(\mu,\delta)}{n}$ et ont démontré que $\{a = 0\} = \{(\mu, \delta) : a(\mu, \delta) = 0\}$ est une courbe de transition de phase continue.

Problème 12.76 *Caractériser la courbe $a(\mu, \delta) = 0$.*

12.7 Alignement multiple

Problème 12.77 *Trouver un algorithme efficace au niveau spatial pour l'alignement multiple.*

Problème 12.78 *Trouver un algorithme qui assemble des alignements multiples à partir d'alignements de niveau 3.*

Problème 12.79 *Construire un exemple pour lequel l'algorithme de multiplication de matrices de Vingron-Argos nécessite $\Omega(L)$ itérations, où L est la longueur des séquences.*

Jiang et Li, 1994 [180] ont formulé le problème suivant :

Problème 12.80 *Les plus courtes (SCS) et les plus longues (LCS) super-séquences communes peuvent-elles être approchées avec un rapport meilleur que* 2 *?*

On peut soutenir que la notion de NP-complétude est quelque peu fallacieuse pour des problèmes de bio-informatique, car elle est insensible aux champs limités des paramètres, qui sont souvent importants en pratique. Par exemple, dans de nombreuses applications, on serait content d'avoir des algorithmes efficaces pour l'alignement multiple avec $k \leq 10$. Ce que nous avons habituellement, c'est l'algorithme de programmation dynamique en $O((2n)^k)$. La NP-complétude du problème de l'alignement multiple ne nous dit quasiment rien sur ce que l'on peut espérer si l'on fixe notre attention sur la gamme des $k \leq 10$. Il se pourrait même qu'il existe un algorithme de complexité temporelle linéaire pour chaque valeur de k fixée ! Par exemple, la NP-complétude ne serait pas contredite si le problème pouvait être résolu en un temps en $O(2^k n)$.

La dernière décennie a vu le développement d'algorithmes particulièrement applicables aux problèmes comme l'alignement multiple pour des éventails de paramètres fixés. Actuellement, on ne sait pas si la complexité obtenue par la programmation dynamique est le « dernier mot » concernant la complexité du problème de l'alignement multiple (Bodlaender *et al.*, 1995 [38]). Mike Fellows a formulé la conjecture suivante :

Problème 12.81 *Le problème de la plus longue sous-séquence commune pour k séquences dans un alphabet de taille fixée peut être résolu en un temps $f(k)n^\alpha$, où α ne dépend pas de k.*

12.8 Trouver des signaux dans l'ADN

Problème 12.82 *Décrire une stratégie gagnante pour B dans le meilleur pari pour les niais.*

Problème 12.83 *Décrire la meilleure stratégie pour A dans le meilleur pari pour les niais (c'est-à-dire la stratégie qui minimise les pertes).*

Dans le meilleur pari pour les niais, si une pièce est truquée (par exemple, $p(0) > p(1)$), il est logique que A choisisse un mot comme $0 \ldots 0$ pour augmenter ses chances.

Problème 12.84 *Étudier le meilleur pari pour les niais avec une pièce truquée. Le joueur B possède-t-il toujours un avantage sur A dans ce cas ?*

Problème 12.85 *Trouver la variance du nombre d'occurrences d'un mot donné dans le cas de chaînes linéaires.*

Problème 12.86 *Trouver un algorithme d'approximation pour le problème du mot consensus.*

Le problème du décodage peut être formulé sous la forme d'un problème du plus long chemin dans un graphe acyclique orienté. Ceci motive une question sur une version efficace au niveau spatial concernant l'algorithme de Viterbi.

Problème 12.87 *Existe-t-il un algorithme de complexité spatiale linéaire pour le problème du décodage ?*

12.9 Prédiction génétique

L'algorithme d'alignement épissé trouve des exons dans l'ADN génomique en utilisant une protéine voisine comme patron. Et si l'échantillon n'était pas une protéine mais un autre ADN génomique (non interprété) ? En particulier, peut-on utiliser un ADN génomique de souris (non annoté) pour prédire les gènes humains ?

Problème 12.88 *Généraliser l'algorithme d'alignement épissé à l'alignement d'une séquence génomique le long d'une autre.*

Problème 12.89 *Généraliser l'approche fondée sur la similitude à la prédiction génétique dans le cas où de multiples protéines similaires sont disponibles.*

Sze et Pevzner, 1997 [332] ont formulé le problème suivant :

Problème 12.90 *Modifier l'algorithme d'alignement épissé pour trouver des alignements épissés suboptimaux.*

Le « jeu des vingt questions avec un menteur » suppose que toute réponse du jeu est fausse avec la probabilité p. Évidemment, si p vaut $\frac{1}{2}$, le jeu est perdu, puisque le menteur ne nous communique aucune information.

Problème 12.91 *Imaginer une stratégie efficace pour le « jeu des vingt questions avec un menteur », qui trouve k entiers inconnus dans l'intervalle $[1, n]$ si la probabilité d'une réponse fausse est $p \neq \frac{1}{2}$.*

Problème 12.92 *Estimer l'espérance du nombre de questions dans le « jeu des vingt questions avec un menteur », si la probabilité d'une réponse fausse est $p \neq \frac{1}{2}$.*

Problème 12.93 *Imaginer des protocoles expérimentaux et informatiques permettant de trouver toutes les variantes d'épissage alternatif pour une séquence génomique donnée.*

L'observation selon laquelle les requêtes PCR peuvent être utilisées pour tester un nombre potentiellement exponentiel d'hypothèses sur les variantes d'épissage aboutit à une reformulation du problème ci-dessus.

Problème 12.94 *Étant donné un graphe $G(V, E)$ avec une collection $\mathcal{C}$ de chemins (inconnus), reconstruire $\mathcal{C}$ en posant le minimum de questions de la forme : « La collection $\mathcal{C}$ contient-elle un chemin qui passe par les sommets v et w de G ? »*

Soit S un ensemble de sondes fixé et soit C une séquence d'ADN. Une empreinte digitale de C est un sous-ensemble de sondes de S qui s'hybrident à C. Soit G une séquence génomique contenant un gène représenté par un clone C d'ADNc. Mironov et Pevzner, 1999 [241] ont étudié le problème de reconnaissance génétique suivant, fondé sur les empreintes digitales :

Problème 12.95 *Étant données une séquence génomique G et l'empreinte digitale du clone C d'ADNc correspondant, prédire un gène dans G (i.e. prédire tous les exons dans G).*

12.10 Réarrangements génomiques

Le tri par inversions correspond à l'élimination des points de rupture. Cependant, pour certaines permutations (comme 563412), aucune inversion ne réduit le nombre de points de rupture. Les trois bandes (intervalles maximaux sans point de rupture) dans 563412 sont croissantes.

Problème 12.96 *Prouver que, si une permutation non signée possède une bande décroissante, il existe une inversion qui réduit le nombre de points de rupture d'au moins une unité.*

Un algorithme 2-glouton pour trier π par inversions choisit des inversions ρ et σ, de sorte que le nombre de points de rupture dans $\pi \cdot \rho \cdot \sigma$ soit minimal parmi toutes les paires d'inversions.

Problème 12.97 *Prouver que l'algorithme 2-glouton présente une garantie de performance pour le tri par inversions.*

Dans le cas où la séquence de gènes contient des duplications, le tri de *permutations* par inversions devient un tri de *mots* par inversions. Par exemple, la plus courte séquence d'inversions pour transformer le mot 43132143 en le mot 42341314 implique deux inversions : 4**3132**143 → 423**13143** → 42341314.

Problème 12.98 *Imaginer un algorithme avec garantie de performance pour trier des mots par inversions.*

Kececioglu et Sankoff, 1994 [193] ont étudié les bornes pour le diamètre $D(n)$ dans le cas de permutations signées ; ils ont prouvé l'encadrement suivant : $n-1 \leq D(n) \leq n$. Ils ont également conjecturé la chose suivante :

Problème 12.99 *Pour des permutations circulaires signées, on a $D(n) = n$ pour n suffisamment grand.*

Problème 12.100 *Caractériser l'ensemble des permutations sur n éléments « difficiles à trier », qui vérifient $d(\pi) = D(n)$.*

Problème 12.101 *Améliorer la borne inférieure et trouver une borne supérieure pour l'espérance de la distance d'inversion.*

Problème 12.102 *Estimer la variance de la distance d'inversion.*

Gates et Papadimitriou, 1979 [120] ont émis la conjecture suivante : « une permutation particulière sur n éléments nécessite au moins $\frac{19}{16}n$ inversions pour être triée. » Heydari et Sudborough, 1997 [161] ont réfuté cette conjecture en décrivant $\frac{18}{16}n + 2$ inversions permettant de trier la permutation de Gates-Papadimitriou.

Problème 12.103 *Trouver le diamètre d'inversion de préfixes du groupe symétrique.*

Les génomes ne se développent pas seulement par des inversions, mais aussi par des *transpositions*. Pour une permutation π, une *transposition* $\rho(i,j,k)$ (définie pour tous $1 \leq i < j \leq n+1$ et tout $1 \leq k \leq n+1$ avec $k \notin [i,j]$) « insère » un intervalle $[i, j-1]$ de π entre π_{k-1} et π_k. Alors $\rho(i,j,k)$ correspond à la permutation

$$\begin{pmatrix} 1 \ldots i-1 & \boxed{\textbf{i i+1 ... ... j-2 j-1}} & \boxed{\text{j ... k-1}} & k \ldots n \\ 1 \ldots i-1 & \boxed{\text{j ... k-1}} & \boxed{\textbf{i i+1 ... ... j-2 j-1}} & k \ldots n \end{pmatrix}$$

Étant données des permutations π et σ, la *distance de transposition* est la longueur de la plus courte série de transpositions $\rho_1, \rho_2, \ldots, \rho_t$ transformant π en $\pi \cdot \rho_1 \cdot \rho_2 \cdots \rho_t = \sigma$. *Trier π par transpositions* revient à trouver la distance de transposition $d(\pi)$ entre π et la permutation identité $\imath$. Bafna et Pevzner, 1998 [20] ont trouvé un algorithme de garantie de performance $1,5$ pour trier par transpositions ; ils ont également démontré que le diamètre de transposition $D_t(n)$ du groupe symétrique vérifiait $\frac{n}{2} \leq D_t(n) \leq \frac{3n}{4}$.

Problème 12.104 *Trouver le diamètre de transposition du groupe symétrique.*

La variante la plus connue du tri par transpositions est le tri par transpositions $\rho(i, i+1, i+2)$, où l'opération est un échange d'éléments adjacents. Un simple algorithme de tri à bulle résout ce problème pour des permutations linéaires. Le cas des permutations circulaires est plus difficile.

Problème 12.105 *Trouver un algorithme pour trier des permutations circulaires par des échanges d'éléments adjacents.*

Problème 12.106 *Toute permutation circulaire peut être triée en* $2\lceil\frac{n-1}{2}\rceil\lfloor\frac{n-1}{2}\rfloor$ *échanges d'éléments adjacents.*

On représente une permutation circulaire par des éléments $\pi_1 \ldots \pi_n$ espacés de façon régulière sur un cercle. La figure 12.2 présente des permutations circulaires $\pi = \pi_1 \ldots \pi_n$ et $\sigma = \sigma_1 \ldots \sigma_n$ positionnées sur deux cercles concentriques et n arêtes $e_1 \ldots e_n$, telles que e_i relie l'élément i dans π et l'élément i dans σ. Une telle représentation de π et σ est appelée un *plongement* ; on s'intéresse aux plongements qui minimisent le nombre C d'arêtes *coupantes.* Dans un plongement, les arêtes peuvent être orientées soit dans le sens des aiguilles d'une montre, soit dans le sens inverse ; notons que le nombre total d'arêtes coupantes dépend du choix des directions. Par exemple, le plongement de gauche de la figure 12.2 correspond à $C = 2$, tandis que celui de droite correspond à $C = 3$. Un vecteur *directeur* de dimension n, $\mathbf{v} = v_1, \ldots v_n$, avec $v_i \in \{+1, -1\}$, définit un plongement en orientant l'arête e_i dans le sens des aiguilles d'une montre si v_i vaut $+1$ et dans le sens inverse sinon.

Pour plus de commodité, on choisit le sommet « midi » d'un cercle pour représenter un point « de départ » d'une permutation circulaire. Choisir un élément r comme point de départ de π définit une *rotation de* π. Dans un souci de simplicité, on suppose que $\sigma = 1 \ldots n$ est la permutation identité et que le point de départ de σ est 1.

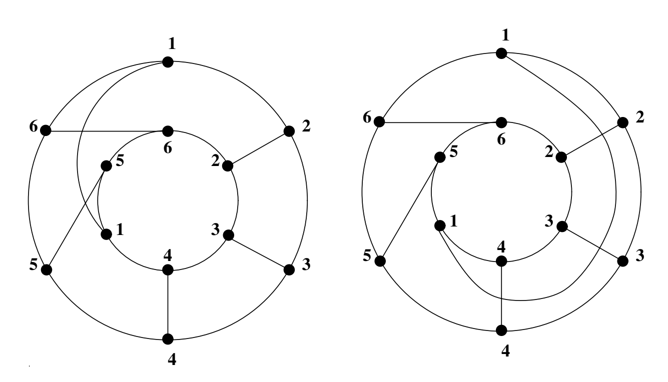


Figure 12.2 – Arêtes coupantes dans des plongements.

Toute rotation de π avec r comme point de départ et tout vecteur $\mathbf{v}$ définissent un plongement ayant un nombre $C(r, \mathbf{v})$ d'arêtes *coupantes.* Sankoff et Goldstein, 1989 [303] ont étudié le problème suivant du *plongement optimal* :

Problème 12.107 *Trouver*

$$\min_{r,\mathbf{v}} C(r, \mathbf{v}).$$

Soit $d_{ij}(\pi)$ la distance entre l'élément i et l'élément j d'une permutation π, orientée dans le sens des aiguilles d'une montre. Par définition, la *longueur* d'une arête e_i orientée dans le même sens est $\hat{\imath} = (d_{1i}(\sigma) - d_{1i}(\pi)) \bmod(n)$, tandis que la longueur d'une arête orientée dans le sens inverse est $\check{\imath} = n - \hat{\imath}$. Pour une rotation r, on définit un vecteur directeur *canonique* $\mathbf{v}(r) = (v_1(r) \dots v_n(r))$ selon la règle suivante : $v_i(r) = +1$ si l'arête e_i qui est dans le sens des aiguilles d'une montre est plus courte que l'arête e_i qui va dans le sens inverse (i.e. si l'on a $\hat{\imath} < \check{\imath}$) et $v_i(r) = -1$ sinon.

Problème 12.108 *Prouver l'inégalité suivante :*

$$\min_r C(r, \mathbf{v}(r)) \leq \lceil \frac{n-1}{2} \rceil \lfloor \frac{n-1}{2} \rfloor.$$

Problème 12.109 *Prouver que, pour tout* $0 \leq r \leq n-1$, *une permutation circulaire* π *peut être triée avec au plus* $C(r, \mathbf{v}(r))$ *échanges d'éléments adjacents.*

Une $\mathcal{A}$-permutation sur n éléments est une permutation de $\{1, 2, \dots, n\}$ intercalée avec des lettres d'un alphabet $\mathcal{A}$. Par exemple, 3aa1baa4a52b est une $\mathcal{A}$-permutation de 31452, avec $\mathcal{A} = \{a, b\}$. Une inversion d'une $\mathcal{A}$-permutation est dite valide si ses éléments de départ et de fin sont égaux. Une $\mathcal{A}$-permutation identité est une permutation dans laquelle $1, 2, \dots, n$ apparaissent dans cet ordre (avec une répartition arbitraire des éléments de $\mathcal{A}$).

Problème 12.110 *Imaginer un test qui décide si une* $\mathcal{A}$*-permutation peut être triée par des inversions valides.*

Problème 12.111 *Proposer un algorithme pour trier des* $\mathcal{A}$*-permutations par des inversions valides.*

12.11 Protéomique informatique

La spectrométrie de masse est devenue une source de nouvelles séquences protéiques, dont certaines étaient auparavant inconnues au niveau de l'ADN. Le problème consiste alors à faire la soudure entre la protéomique et la génomique, i.e. le problème du séquençage de l'ADN basé sur l'information obtenue grâce au séquençage peptidique MS/MS à grande échelle. Ce problème est compliqué, car les peptides séquencés par spectrométrie de masse peuvent être de petits morceaux avec de possibles ambiguïtés (telles que la transposition d'acides aminés adjacents).

Problème 12.112 *Étant donné un ensemble de peptides (avec ambiguïtés) d'une protéine donnée, imaginer des protocoles expérimentaux et informatiques pour trouver la séquence génomique qui correspond à cette protéine.*

Problème 12.113 *Imaginer un algorithme pour trouver des peptides avec ambiguïtés dans des bases de données protéiques.*

Considérons un mélange de protéines (inconnues) soumis à une digestion complète par une protéase (la trypsine, par exemple). Ceci aboutit à une collection de peptides dont la longueur varie de 10 à 20 acides aminés ; le problème consiste alors à décider quels peptides appartiennent aux mêmes protéines, afin de reconstruire l'ordre des peptides dans chacune d'elles. Le spectromètre de masse est capable de séquencer (partiellement) tous les peptides du mélange, mais on ne sait pas habituellement quels peptides viennent de la même protéine, ni quel est l'ordre des peptides dans ces protéines. Le séquençage protéique de mélanges de protéines est un problème d'assemblage de peptides en protéines individuelles.

Problème 12.114 *Imaginer des protocoles expérimentaux et informatiques pour le séquençage de mélanges protéiques par spectrométrie de masse.*

L'approche qui utilise le graphe spectral pour le séquençage peptidique *de novo* ne prend en compte ni les ions internes ni les ions à charges multiples.

Problème 12.115 *Proposer un algorithme de séquençage peptidique qui prend en compte les ions internes et les ions à charges multiples.*

Soit $M(P)$ l'ensemble des masses de tous les peptides partiels d'un peptide P. À l'aide de la digestion de P par différentes protéases non spécifiques, on peut obtenir un ensemble de masses, mesurées expérimentalement, des peptides partiels $M \subset M(P)$.

Problème 12.116 *Étant donné un ensemble de masses $M \subset M(P)$, reconstruire P.*

Une détermination précise de la masse du peptide parent est extrêmement importante dans le séquençage peptidique *de novo*. Une erreur dans la masse parente amène des erreurs systématiques dans les masses des sommets pour les ions C-terminaux, rendant ainsi difficile la reconstruction peptidique. En pratique, les écarts entre les masses peptidiques réelles (données par la somme d'acides aminés d'un peptide) et les masses parentes observées expérimentalement sont souvent si grands que les erreurs dans la reconstruction peptidique deviennent presque inévitables.

Problème 12.117 *Étant donné un spectre MS/MS (sans masse parente), imaginer un algorithme qui estime la masse parente.*

Annexe : introduction à la biologie moléculaire

Évidemment, on ne trouvera pas ici l'intégralité de ce qu'il faut connaître ; pour une introduction plus complète, se reporter à 1999 [220].

L'*ADN* est une chaîne dans l'alphabet à quatre lettres des *nucléotides* A, T, G et C. L'intégralité de l'ADN d'un organisme vivant est appelé son *génome*. Les organismes vivants (comme les humains) ont des milliers de milliards de cellules et chacune d'elles contient le même génome. En longueur, l'ADN varie de quelques millions de lettres (pour les bactéries) à quelques milliards (pour les mammifères). L'ADN forme une hélice, mais ce n'est pas réellement important dans le cadre de ce livre. Ce qui est primordial, c'est que l'ADN est habituellement de l'ADN double brin ; l'un des deux brins est le *complément de Watson-Crick* de l'autre (T est apparié à A et C à G), de la façon suivante :

$$\begin{array}{ccccccccc} A & T & G & C & T & C & A & G & G \\ | & | & | & | & | & | & | & | & | \\ T & A & C & G & A & G & T & C & C \end{array}$$

L'ADN fabrique les éléments polyvalents de la cellule que sont les *protéines*. Les protéines sont de petites chaînes dans l'alphabet à vingt lettres des *acides aminés*. Le génome humain fabrique à peu près 100 000 protéines, chacune d'elles étant constituée de quelques centaines d'acides aminés environ. Les bactéries, quant à elles, fabriquent de 500 à 1500 protéines, ce qui correspond quasiment au minimum vital pour un organisme. Les protéines sont fabriquées par des fragments d'ADN appelés *gènes*, qui sont à peu près trois fois plus longs que les protéines correspondantes. Pourquoi trois ? Parce que chaque triplet de nucléotides dans l'alphabet de l'ADN code une lettre dans l'alphabet protéique des acides aminés. Il existe $4^3 = 64$ triplets (*codons*) et la question qui se pose est la suivante : pourquoi la nature a-t-elle besoin d'autant de combinaisons pour coder vingt acides aminés ? Eh bien, le code génétique (Figure 12.3) est redondant ; de plus, il existe des codons *Stop* signalant la fin de la protéine.

Les biologistes divisent le monde des organismes en deux catégories : les *eucaryotes* (leur ADN est à l'intérieur d'un noyau) et les *procaryotes*. En général, un génome eucaryote n'est pas constitué d'une seule chaîne (comme chez

Seconde position

Première position	T		C		A		G	
T	TTT TTC	**PHE**	TCT TCC TCA TCG	**SER**	TAT TAC	**TYR**	TGT TGC	**CYS**
	TTA TTG	**LEU**			TAA TAG	**Stop**	TGA	**Stop**
							TGG	**TRP**
C	CTT CTC CTA CTG	**LEU**	CCT CCC CCA CCG	**PRO**	CAT CAC	**HIS**	CGT CGC CGA CGG	**ARG**
					CAA CAG	**GLN**		
A	ATT ATC ATA	**ILE**	ACT ACC ACA ACG	**THR**	AAT AAC	**ASN**	AGT AGC	**SER**
	ATG	**MET**			AAA AAG	**LYS**	AGA AGG	**ARG**
G	GTT GTC GTA GTG	**VAL**	GCT GCC GCA GCG	**ALA**	GAT GAC	**ASP**	GGT GGC GGA GGG	**GLY**
					GAA GAG	**GLU**		

Figure 12.3 – Le code génétique.

les procaryotes), mais plutôt d'un ensemble de chaînes appelées des *chromosomes*. Dans notre cas, il faut garder à l'esprit une différence majeure entre procaryotes et eucaryotes : dans les gènes procaryotes, les chaînes sont continues, tandis qu'ils sont cassés en plusieurs morceaux (appelés *exons*) chez les eucaryotes. Un gène humain peut être cassé jusqu'en cinquante exons, séparés par des morceaux apparemment sans signification appelés des *introns*.

Un gène cassé en plusieurs morceaux doit encore produire la protéine correspondante. Pour cela, les cellules doivent découper les introns et assembler les exons. Ceci est réalisé dans l'*ARNm*, une molécule intermédiaire semblable à un ADN court à simple brin, lors d'un procédé appelé *transcription*. Il existe des signaux dans l'ADN pour commencer la transcription, appelés *promoteurs*. Le mécanisme de synthèse protéique *traduit* alors les codons dans l'ARNm en une chaîne d'acides aminés (protéine). En laboratoire, l'ARNm peut également être utilisé comme patron pour fabriquer une copie complémentaire appelée *ADNc*, identique au gène original avec des introns.

Au fil des ans, les biologistes ont appris à faire de nombreuses choses avec l'ADN. Ils ont également découvert comment copier l'ADN en grande quantité pour une étude approfondie. Une façon de faire, la *PCR* (réaction en chaîne de la polymérase), est la presse d'imprimerie de Gutenberg de l'ADN. La PCR amplifie un fragment d'ADN court (100 à 500 nucléotides) et produit un grand nombre de chaînes identiques. Pour utiliser la PCR, il faut connaître une paire de courtes chaînes (20 à 30 lettres) soutenant l'endroit qui nous intéresse, et

créer deux *amorces PCR*, des fragments d'ADN synthétique identiques à ces chaînes. Pourquoi avons-nous besoin d'un grand nombre de courts fragments d'ADN identiques ? D'un point de vue informatique, le fait d'avoir la même chaîne en 10^{18} exemplaires ne signifie pas grand chose ; cela n'accroît pas la somme d'information. Cela signifie cependant beaucoup pour les biologistes, puisque la plupart des expériences biologiques nécessitent un grand nombre de chaînes. Par exemple, la PCR peut être utilisée pour détecter l'existence d'un certain fragment dans un échantillon d'ADN.

Une autre façon de copier l'ADN est de le *cloner*. Contrairement à la PCR, le clonage ne nécessite aucune information préalable sur les amorces soutenantes. Cependant, avec le clonage, les biologistes n'ont aucun contrôle sur le fragment d'ADN qu'ils amplifient. Le procédé commence habituellement par le brisement de l'ADN en petits morceaux. Pour en étudier un en particulier, les biologistes obtiennent de nombreuses copies identiques de chaque morceau en les *clonant*. Le clonage incorpore un fragment d'ADN dans un *vecteur de clonage*. Un vecteur de clonage est une molécule d'ADN (provenant habituellement d'un virus ou de l'ADN d'un organisme supérieur), dans lequel un autre fragment d'ADN peut être inséré. Lors de cette opération, le vecteur de clonage ne perd pas sa faculté d'auto-réplication. Les vecteurs introduisent de l'ADN étranger dans les cellules-hôtes (comme les bactéries), où ils peuvent être reproduits en grande quantité. Le procédé d'auto-réplication crée un grand nombre de copies du fragment, permettant ainsi à sa structure d'être étudiée. Un fragment reproduit de cette façon est appelé un *clone*. Les biologistes peuvent faire des *banques de clones*, constituées de milliers de clones de la même molécule d'ADN (chacun représentant un court fragment d'ADN choisi au hasard).

Les *enzymes de restriction* sont des ciseaux moléculaires qui coupent l'ADN à chaque occurrence de certains mots. Par exemple, l'enzyme de restriction *Bam*HI coupe l'ADN en *fragments de restriction* à chaque occurrence du mot GGATCC. Les protéines peuvent également être coupées en courts fragments (appelés *peptides*) par un autre genre de ciseaux, des *protéases*.

Le procédé qui consiste à joindre deux brins d'ADN complémentaires en une molécule double brin est appelé *hybridation*. L'hybridation d'une courte *sonde* complémentaire à un fragment d'ADN connu peut être utilisée pour détecter la présence de ce fragment d'ADN. Une sonde est un court fragment d'ADN à un seul brin avec un marquage fluorescent ; elle est utilisée pour détecter la présence éventuelle d'une séquence complémentaire dans un échantillon d'ADN. Pourquoi avons-nous besoin de marquer de façon fluorescente la sonde ? Si une sonde s'hybride à un fragment d'ADN, alors nous pouvons le voir à l'aide d'un détecteur spectroscopique.

L'électrophorèse sur gel est une technique qui permet aux biologistes de mesurer la taille des fragments d'ADN sans les découper. L'ADN est une molécule à charge négative qui migre vers un pôle positif dans un champ électrique. La vitesse de migration dépend de la taille du fragment ; par conséquent, la mesure des distances de migration permet aux biologistes d'estimer les tailles des fragments d'ADN.

Bibliographie

[1] A.V. Aho and M.J. Corasick. Efficient string matching : an aid to bibliographic search. *Communication of ACM*, 18 :333–340, 1975.

[2] D.J. Aldous and P. Diaconis. Hammersley's interacting particle process and longest increasing subsequences. *Probability Theory and Related Fields*, 103 :199–213, 1995.

[3] F. Alizadeh, R.M. Karp, L.A. Newberg, and D.K. Weisser. Physical mapping of chromosomes : A combinatorial problem in molecular biology. *Algorithmica*, 13 :52–76, 1995.

[4] F. Alizadeh, R.M. Karp, D.K. Weisser, and G. Zweig. Physical mapping of chromosomes using unique probes. *Journal of Computational Biology*, 2 :159–184, 1995.

[5] S. Altschul, W. Gish, W. Miller, E. Myers, and J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215 :403–410, 1990.

[6] S.F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *Journal of Molecular Biology*, 219 :555–565, 1991.

[7] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped Blast and Psi-Blast : a new generation of protein database search programs. *Nucleic Acids Research*, 25 :3389–3402, 1997.

[8] T.S. Anantharaman, B. Mishra, and D.C. Schwartz. Genomics via optical mapping. II : Ordered restriction maps. *Journal of Computational Biology*, 4 :91–118, 1997.

[9] A. Apostolico. Improving the worst-case performance of the Hunt-Szymanski strategy for the longest common subsequence of two strings. *Information Processing Letters*, 23 :63–69, 1986.

[10] A. Apostolico and F. Preparata. Data structures and algorithms for the string statistics problem. *Algorithmica*, 15 :481–494, 1996.

[11] R. Arratia, E.S. Lander, S. Tavare, and M.S. Waterman. Genomic mapping by anchoring random clones : a mathematical analysis. *Genomics*, 11 :806–827, 1991.

[12] R. Arratia, D. Martin, G. Reinert, and M.S. Waterman. Poisson process approximation for sequence repeats, and sequencing by hybridization. *Journal of Computational Biology*, 3 :425–464, 1996.

[13] R. Arratia and M.S. Waterman. The Erdös-Rényi strong law for pattern matching with a given proportion of mismatches. *Annals of Probability*, 17 :1152–1169, 1989.

[14] R. Arratia and M.S. Waterman. A phase transition for the score in matching random sequences allowing deletions. *Annals of Applied Probability*, 4 :200–225, 1994.

[15] R. Baer and P. Brock. Natural sorting over permutation spaces. *Math. Comp.*, 22 :385–410, 1968.

[16] R.A. Baeza-Yates and G.H. Gonnet. A new approach to text searching. In *Proceedings of the Twelfth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 168–175, Cambridge, Massachussets, 1989.

[17] R.A. Baeza-Yates and C.H. Perleberg. Fast and practical approximate string matching. In *Third Annual Symposium on Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 185–192, Tucson, Arizona, April/May 1992. Springer-Verlag.

[18] V. Bafna, E.L. Lawler, and P.A. Pevzner. Approximation algorithms for multiple sequence alignment. *Theoretical Computer Science*, 182 :233–244, 1997.

[19] V. Bafna and P.A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25 :272–289, 1996.

[20] V. Bafna and P.A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11 :224–240, 1998.

[21] J. Baik, P.A. Deift, and K. Johansson. On the distribution of the length of the longest subsequence of random permutations. *Journal of the American Mathematical Society*, 12 :1119–1178, 1999.

[22] W. Bains. Multan : a program to align multiple DNA sequences. *Nucleic Acids Research*, 14 :159–177, 1986.

[23] W. Bains and G. Smith. A novel method for nucleic acid sequence determination. *Journal of Theoretical Biology*, 135 :303–307, 1988.

[24] P. Baldi and S. Brunak. *Bioinformatics : The Machine Learning Approach*. The MIT Press, 1997.

[25] E. Barillot, B. Lacroix, and D. Cohen. Theoretical analysis of library screening using an N-dimensional pooling strategy. *Nucleic Acids Research*, 19 :6241–6247, 1991.

[26] C. Bartels. Fast algorithm for peptide sequencing by mass spectroscopy. *Biomedical and Environmental Mass Spectrometry*, 19 :363–368, 1990.

[27] G.J. Barton and M.J.E. Sternberg. A strategy for the rapid multiple alignment of protein sequences. *Journal of Molecular Biology*, 198 :327–337, 1987.

[28] A. Baxevanis and B.F. Ouellette. *Bioinformatics : A Practical Guide to the Analysis of Genes and Proteins.* Wiley-Interscience, 1998.

[29] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[30] G. Benson. Sequence alignment with tandem duplication. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97)*, pages 27–36, Santa Fe, New Mexico, January 1997. ACM Press.

[31] G. Benson. An algorithm for finding tandem repeats of unspecified pattern size. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB-98)*, pages 20–29, New York, New York, March 1998. ACM Press.

[32] S.M. Berget, C. Moore, and P.A. Sharp. Spliced segments at the 5' terminus of adenovirus 2 late mRNA. *Proceedings of the National Academy of Sciences USA*, 74 :3171–3175, 1977.

[33] P. Berman and S. Hannenhalli. Fast sorting by reversal. In *Seventh Annual Symposium on Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 168–185, Laguna Beach, California, June 1996. Springer-Verlag.

[34] P. Berman, Z. Zhang, Y.I. Wolf, E.V. Koonin, and W. Miller. Winnowing sequences from a database search. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB-99)*, pages 50–58, New York, New York, March 1999. ACM Press.

[35] K. Biemann and H.A. Scoble. Characterization of tandem mass spectrometry of structural modifications in proteins. *Science*, 237 :992–998, 1987.

[36] B.E. Blaisdell. A measure of the similarity of sets of sequences not requiring sequence alignment. *Proceedings of the National Academy of Sciences USA*, 16 :5169–5174, 1988.

[37] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *Journal of the ACM*, 41 :630–647, 1994.

[38] H.L. Bodlaender, R.G. Downey, M.R. Fellows, and H.T. Wareham. The parameterized complexity of sequence alignment and consensus. *Theoretical Computer Science*, 147 :31–54, 1995.

[39] M. Boehnke, K. Lange, and D.R. Cox. Statistical methods for multipoint radiation hybrid mapping. *American Journal of Human Genetics*, 49 :1174–1188, 1991.

[40] K.S. Booth and G.S. Leuker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13 :335–379, 1976.

[41] P. Bork and T.J. Gibson. Applying motif and profile searches. *Methods in Enzymology*, 266 :162–184, 1996.

[42] M. Borodovsky and J. McIninch. Recognition of genes in DNA sequences with ambiguities. *BioSystems*, 30 :161–171, 1993.

[43] M. Yu. Borodovsky, Yu.A. Sprizhitsky, E.I. Golovanov, and A.A. Alexandrov. Statistical features in the *E. coli* genome functional domains primary structure III. Computer recognition of protein coding regions. *Molecular Biology*, 20 :1144–1150, 1986.

[44] D. Botstein, R.L. White, M. Skolnick, and R.W. Davis. Construction of a genetic linkage map in man using restriction fragment length polymorphisms. *American Journal of Human Genetics*, 32 :314–331, 1980.

[45] R.S. Boyer and J.S. Moore. A fast string searching algorithm. *Communication of ACM*, 20 :762–772, 1977.

[46] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5 :279–305, 1998.

[47] V. Brendel, J.S. Beckman, and E.N. Trifonov. Linguistics of nucleotide sequences : morphology and comparison of vocabularies. *Journal of Biomolecular Structure and Dynamics*, 4 :11–21, 1986.

[48] D. Breslauer, T. Jiang, and Z. Jiang. Rotations of periodic strings and short superstrings. *Journal of Algorithms*, 24 :340–353, 1997.

[49] N. Broude, T. Sano, C. Smith, and C. Cantor. Enhanced DNA sequencing by hybridization. *Proceedings of the National Academy of Sciences USA*, 91 :3072–3076, 1994.

[50] S. Brunak, J. Engelbrecht, and S. Knudsen. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology*, 220 :49–65, 1991.

[51] W.J. Bruno, E. Knill, D.J. Balding, D.C. Bruce, N.A. Doggett, W.W. Sawhill, R.L. Stallings, C.C. Whittaker, and D.C. Torney. Efficient pooling designs for library screening. *Genomics*, 26 :21–30, 1995.

[52] R. Bundschuh and T. Hwa. An analytic study of the phase transition line in local sequence alignment with gaps. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB-99)*, pages 70–76, Lyon, France, April 1999. ACM Press.

[53] C. Burge, A.M. Campbell, and S. Karlin. Over- and under-representation of short oligonucleotides in DNA sequences. *Proceedings of the National Academy of Sciences USA*, 89 :1358–1362, 1992.

[54] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268 :78–94, 1997.

[55] S. Burkhardt, A. Crauser, P. Ferragina, H.-P. Lenhof, E. Rivals, and M. Vingron. q-gram based database searching using a suffix array. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB-99)*, pages 77–83, Lyon, France, April 1999. ACM Press.

[56] A. Caprara. Formulations and complexity of multiple sorting by reversals. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB-99)*, pages 84–93, Lyon, France, April 1999. ACM Press.

[57] A. Caprara. Sorting by reversals is difficult. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97)*, pages 75–83, Santa Fe, New Mexico, January 1997. ACM Press.

[58] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48 :1073–1082, 1988.

[59] R.P. Carstens, J.V. Eaton, H.R. Krigman, P.J. Walther, and M.A. Garcia-Blanco. Alternative splicing of fibroblast growth factor receptor 2 (FGF-R2) in human prostate cancer. *Oncogene*, 15 :3059–3065, 1997.

[60] W.K. Cavenee, M.F. Hansen, M. Nordenskjold, E. Kock, I. Maumenee, J.A. Squire, R.A. Phillips, and B.L. Gallie. Genetic origin of mutations predisposing to retinoblastoma. *Science*, 228 :501–503, 1985.

[61] W.I. Chang and J. Lampe. Theoretical and empirical comparisons of approximate string matching algorithms. In *Third Annual Symposium on Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 175–184, Tucson, Arizona, April/May 1992. Springer-Verlag.

[62] W.I. Chang and E.L. Lawler. Sublinear approximate string matching and biological applications. *Algorithmica*, 12 :327–344, 1994.

[63] K.M. Chao. Computing all suboptimal alignments in linear space. In *Fifth Annual Symposium on Combinatorial Pattern Matching*, volume 807 of *Lecture Notes in Computer Science*, pages 31–42, Asilomar, California, 1994. Springer-Verlag.

[64] K.M. Chao, W.R. Pearson, and W. Miller. Aligning two sequences within a specified diagonal band. *Computer Applications in Biosciences*, 8 :481–487, 1992.

[65] M. Chee, R. Yang, E. Hubbel, A. Berno, X.C. Huang, D. Stern, J. Winkler, D.J. Lockhart, M.S. Morris, and S.P.A. Fodor. Accessing genetic information with high density DNA arrays. *Science*, 274 :610–614, 1996.

[66] A. Chetverin and F. Kramer. Sequencing of pools of nucleic acids on oligonucleotide arrays. *BioSystems*, 30 :215–232, 1993.

[67] L.T. Chow, R.E. Gelinas, T.R. Broker, and R.J. Roberts. An amazing sequence arrangement at the 5' ends of adenovirus 2 messenger RNA. *Cell*, 12 :1–8, 1977.

[68] I. Chumakov, P. Rigault, S. Guillou, P. Ougen A. Billaut, G. Guasconi, P. Gervy, I. LeGall, P. Soularue, and L. Grinas et al. Continuum of overlapping clones spanning the entire human chromosome 21q. *Nature*, 359 :380–387, 1992.

[69] G. Churchill. Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology*, 51 :79–94, 1989.

[70] V. Chvátal and D. Sankoff. Longest common subsequences of two random sequences. *Journal of Applied Probability*, 12 :306–315, 1975.

[71] V. Chvatal and D. Sankoff. An upper-bound techniques for lengths of common subsequences. In D. Sankoff and J.B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules : The Theory and Practice of Sequence Comparison*, pages 353–357. Addison-Wesley, 1983.

[72] K.R. Clauser, P.R. Baker, and A.L. Burlingame. The role of accurate mass measurement (+/– 10ppm) in protein identification strategies employing MS or MS/MS and database searching. *Analytical Chemistry*, 71 :2871–2882, 1999.

[73] F.S. Collins, M.L. Drumm, J.L. Cole, W.K. Lockwood, G.F. Vande Woude, and M.C. Iannuzzi. Construction of a general human chromosome jumping library, with application to cystic fibrosis. *Science*, 235 :1046–1049, 1987.

[74] N.G. Copeland, N.A. Jenkins, D.J. Gilbert, J.T. Eppig, L.J. Maltals, J.C. Miller, W.F. Dietrich, A. Weaver, S.E. Lincoln, R.G. Steen, L.D. Steen, J.H. Nadeau, and E.S. Lander. A genetic linkage map of the mouse : Current applications and future prospects. *Science*, 262 :57–65, 1993.

[75] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. The MIT Press, 1989.

[76] A. Coulson, J. Sulston, S. Brenner, and J. Karn. Toward a physical map of the genome of the nematode, *Caenorhabditis elegans*. *Proceedings of the National Academy of Sciences USA*, 83 :7821–7825, 1986.

[77] D.R. Cox, M. Burmeister, E.R. Price, S. Kim, and R.M. Myers. Radiation hybrid mapping : a somatic cell genetic method for constructing high-resolution maps of mammalian chromosomes. *Science*, 250 :245–250, 1990.

[78] E. Czabarka, G. Konjevod, M. Marathe, A. Percus, and D.C. Torney. Algorithms for optimizing production DNA sequencing. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000)*, pages 399–408, San Francisco, California, 2000. SIAM Press.

[79] V. Dancik, T. Addona, K. Clauser, J. Vath, and P.A. Pevzner. De novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 6 :327–342, 1999.

[80] K.J. Danna, G.H. Sack, and D. Nathans. Studies of simian virus 40 DNA. VII. a cleavage map of the SV40 genome. *Journal of Molecular Biology*, 78 :263–276, 1973.

[81] K.E. Davies, P.L. Pearson, P.S. Harper, J.M. Murray, T. O'Brien, M. Sarfarazi, and R. Williamson. Linkage analysis of two cloned DNA sequences flanking the Duchenne muscular dystrophy locus on the short arm of the human X chromosome. *Nucleic Acids Research*, 11 :2303–2312, 1983.

[82] M.A. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure*, chapter 5, pages 345–352. 1978.

[83] J. Deken. Some limit results for longest common subsequences. *Discrete Mathematics*, 26 :17–31, 1979.

[84] J. Deken. Probabilistic behavior of longest common subsequence length. In D. Sankoff and J.B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules : The Theory and Practice of Sequence Comparison*, pages 359–362. Addison-Wesley, 1983.

[85] A. Dembo and S. Karlin. Strong limit theorem of empirical functions for large exceedances of partial sums of i.i.d. variables. *Annals of Probability*, 19 :1737–1755, 1991.

[86] R.P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51 :161–165, 1950.

[87] T. Dobzhansky and A.H. Sturtevant. Inversions in the chromosomes of *Drosophila pseudoobscura*. *Genetics*, 23 :28–64, 1938.

[88] H. Donis-Keller, P. Green, C. Helms, S. Cartinhour, B. Weiffenbach, K. Stephens, T.P. Keith, D.W. Bowden, D.R. Smith, and E.S. Lander. A genetic linkage map of the human genome. *Cell*, 51 :319–337, 1987.

[89] R.F. Doolittle, M.W. Hunkapiller, L.E. Hood, S.G. Devare, K.C. Robbins, S.A. Aaronson, and H.N. Antoniades. Simian sarcoma virus onc gene, v-sis, is derived from the gene (or genes) encoding a platelet-derived growth factor. *Science*, 221 :275–277, 1983.

[90] R. Drmanac, S. Drmanac, Z. Strezoska, T. Paunesku, I. Labat, M. Zeremski, J. Snoddy, W.K. Funkhouser, B. Koop, and L. Hood. DNA sequence determination by hybridization : a strategy for efficient large-scale sequencing. *Science*, 260 :1649–1652, 1993.

[91] R. Drmanac, I. Labat, I. Brukner, and R. Crkvenjakov. Sequencing of megabase plus DNA by hybridization : theory of the method. *Genomics*, 4 :114–128, 1989.

[92] J. Dumas and J. Ninio. Efficient algorithms for folding and comparing nucleic acid sequences. *Nucleic Acids Research*, 10 :197–206, 1982.

[93] R. Durbin, S. Eddy, A. Krogh, and G. Mitchinson. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[94] M. Dyer, A. Frieze, and S. Suen. The probability of unique solutions of sequencing by hybridization. *Journal of Computational Biology*, 1 :105–110, 1994.

[95] S.R. Eddy and R. Durbin. RNA sequence analysis using covariance models. *Nucleic Acids Research*, 22 :2079–2088, 1994.

[96] N. El-Mabrouk, D. Bryant, and D. Sankoff. Reconstructing the predoubling genome. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB-99)*, pages 154–163, Lyon, France, April 1999. ACM Press.

[97] J. Eng, A. McCormack, and J. Yates. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of American Society for Mass Spectometry*, 5 :976–989, 1994.

[98] G.A. Evans and K.A. Lewis. Physical mapping of complex genomes by cosmid multiplex analysis. *Proceedings of the National Academy of Sciences USA*, 86 :5030–5034, 1989.

[99] W. Feldman and P.A. Pevzner. Gray code masks for sequencing by hybridization. *Genomics*, 23 :233–235, 1994.

[100] D. Feng and R. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25 :351–360, 1987.

[101] D. Fenyo, J. Qin, and B.T. Chait. Protein identification using mass spectrometric information. *Electrophoresis*, 19 :998–1005, 1998.

[102] D. Fernandez-Baca, T. Seppalainen, and G. Slutzki. Bounds for parametric sequence comparison. In *Sixth International Symposium on String Processing and Information Retrieval*, pages 55–62, Cancun, Mexico, September 1999. IEEE Computer Society.

[103] J. Fernández-de Cossío, J. Gonzales, and V. Besada. A computer program to aid the sequencing of peptides in collision-activated decomposition experiments. *Computer Applications in Biosciences*, 11 :427–434, 1995.

[104] J.W. Fickett. Recognition of protein coding regions in DNA sequences. *Nucleic Acids Research*, 10 :5303–5318, 1982.

[105] J.W. Fickett. Finding genes by computer : the state of the art. *Trends in Genetics*, 12 :316–320, 1996.

[106] J.W. Fickett and C.S. Tung. Assessment of protein coding measures. *Nucleic Acids Research*, 20 :6441–6450, 1992.

[107] W.M. Fitch and T.F. Smith. Optimal sequence alignments. *Proceedings of the National Academy of Sciences USA*, 80 :1382–1386, 1983.

[108] H. Fleischner. *Eulerian Graphs and Related Topics.* Elsevier Science Publishers, 1990.

[109] S.P.A. Fodor, R.P. Rava, X.C. Huang, A.C. Pease, C.P. Holmes, and C.L. Adams. Multiplex biochemical assays with biological chips. *Nature*, 364 :555–556, 1993.

[110] S.P.A. Fodor, J.L. Read, M.S. Pirrung, L. Stryer, A.T. Lu, and D. Solas. Light-directed spatially addressable parallel chemical synthesis. *Science*, 251 :767–773, 1991.

[111] S. Foote, D. Vollrath, A. Hilton, and D.C. Page. The human Y chromosome : overlapping DNA clones spanning the euchromatic region. *Science*, 258 :60–66, 1992.

[112] D. Fousler and S. Karlin. Maximum success duration for a semi-markov process. *Stochastic Processes and their Applications*, 24 :203–210, 1987.

[113] D. Frishman, A. Mironov, H.W. Mewes, and M.S. Gelfand. Combining diverse evidence for gene recognition in completely sequenced bacterial genomes. *Nucleic Acids Research*, 26 :2941–2947, 1998.

[114] D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15 :835–856, 1965.

[115] D.J. Galas, M. Eggert, and M.S. Waterman. Rigorous pattern-recognition methods for DNA sequences. Analysis of promoter sequences from Escherichia coli. *Journal of Molecular Biology*, 186 :117–128, 1985.

[116] Z. Galil and R. Giancarlo. Speeding up dynamic programming with applications to molecular biology. *Theoretical Computer Science*, 64 :107–118, 1989.

[117] J. Gallant, D. Maier, and J. Storer. On finding minimal length superstrings. *Journal of Computer and System Science*, 20 :50–58, 1980.

[118] M. Gardner. On the paradoxial situations that arise from nontransitive relationships. *Scientific American*, pages 120–125, October 1974.

[119] M.R. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., 1979.

[120] W.H. Gates and C.H. Papadimitriou. Bounds for sorting by prefix reversals. *Discrete Mathematics*, 27 :47–57, 1979.

[121] M.S. Gelfand. Computer prediction of exon-intron structure of mammalian pre-mRNAs. *Nucleic Acids Research*, 18 :5865–5869, 1990.

[122] M.S. Gelfand. Statistical analysis and prediction of the exonic structure of human genes. *Journal of Molecular Evolution*, 35 :239–252, 1992.

[123] M.S. Gelfand. Prediction of function in DNA sequence analysis. *Journal of Computational Biology*, 2 :87–115, 1995.

[124] M.S. Gelfand and E.V. Koonin. Avoidance of palindromic words in bacterial and archaeal genomes : a close connection with restriction enzymes. *Nucleic Acids Research*, 25 :2430–2439, 1997.

[125] M.S. Gelfand, A.A. Mironov, and P.A. Pevzner. Gene recognition via spliced sequence alignment. *Proceedings of the National Academy of Sciences USA*, 93 :9061–9066, 1996.

[126] J.F. Gentleman and R.C. Mullin. The distribution of the frequency of occurrence of nucleotide subsequences, based on their overlap capability. *Biometrics*, 45 :35–52, 1989.

[127] W. Gillett, J. Daues, L. Hanks, and R. Capra. Fragment collapsing and splitting while assembling high-resolution restriction maps. *Journal of Computational Biology*, 2 :185–205, 1995.

[128] P.C. Gilmore and A.J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16 :539–548, 1964.

[129] W. Gish and D.J. States. Identification of protein coding regions by database similarity search. *Nature Genetics*, 3 :266–272, 1993.

[130] L. Goldstein and M.S. Waterman. Mapping DNA by stochastic relaxation. *Advances in Applied Mathematics*, 8 :194–207, 1987.

[131] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer : class discovery and class prediction by gene expression monitoring. *Science*, 286 :531–537, 1999.

[132] M. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[133] G.H. Gonnet, M.A. Cohen, and S.A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256 :1443–1445, 1992.

[134] A. Gooley and N. Packer. The importance of co- and post-translational modifications in proteome projects. In W. Wilkins, K. Williams, R. Appel, and D. Hochstrasser, editors, *Proteome Research : New Frontiers in Functional Genomics*, pages 65–91. Springer-Verlag, 1997.

[135] O. Gotoh. Consistency of optimal sequence alignments. *Bulletin of Mathematical Biology*, 52 :509–525, 1990.

[136] P. Green. Documentation for phrap. http ://bozeman.mbt.washington.edu/phrap.docs/phrap.html.

[137] D.S. Greenberg and S. Istrail. Physical mapping by STS hybridization : algorithmic strategies and the challenge of software evaluation. *Journal of Computational Biology*, 2 :219–273, 1995.

[138] M. Gribskov, J. Devereux, and R.R. Burgess. The codon preference plot : graphic analysis of protein coding sequences and prediction of gene expression. *Nucleic Acids Research*, 12 :539–549, 1984.

[139] M. Gribskov, M. McLachlan, and D. Eisenberg. Profile analysis : detection of distantly related proteins. *Proceedings of the National Academy of Sciences USA*, 84 :4355–4358, 1987.

[140] R. Grossi and F. Luccio. Simple and efficient string matching with k mismatches. *Information Processing Letters*, 33 :113–120, 1989.

[141] L.J. Guibas and A.M. Odlyzko. String overlaps, pattern matching and nontransitive games. *Journal of Combinatorial Theory, Series A*, 30 :183–208, 1981.

[142] R. Guigo, S. Knudsen, N. Drake, and T.F. Smith. Prediction of gene structure. *Journal of Molecular Biology*, 226 :141–157, 1992.

[143] J.F. Gusella, N.S. Wexler, P.M. Conneally, S.L. Naylor, M.A. Anderson, R.E. Tanzi, P.C. Watkins, K. Ottina, M.R. Wallace, A.Y. Sakaguchi, A.B. Young, I. Shoulson, E. Bonilla, and J.B. Martin. A polymorphic DNA marker genetically linked to Huntington's disease. *Nature*, 306 :234–238, 1983.

[144] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55 :141–154, 1993.

[145] D. Gusfield. *Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology.* Cambridge University Press, 1997.

[146] D. Gusfield, K. Balasubramanian, and D. Naor. Parametric optimization of sequence alignment. *Algorithmica*, 12 :312–326, 1994.

[147] D. Gusfield, R. Karp, L. Wang, and P. Stelling. Graph traversals, genes and matroids : An efficient case of the travelling salesman problem. *Discrete Applied Mathematics*, 88 :167–180, 1998.

[148] J.G. Hacia, J.B. Fan, O. Ryder, L. Jin, K. Edgemon, G. Ghandour, R.A. Mayer, B. Sun, L. Hsie, C.M. Robbins, L.C. Brody, D. Wang, E.S. Lander, R. Lipshutz, S.P. Fodor, and F.S. Collins. Determination of ancestral alleles for human single-nucleotide polymorphisms using high-density oligonucleotide arrays. *Nature Genetics*, 22 :164–167, 1999.

[149] C.W. Hamm, W.E. Wilson, and D.J. Harvan. Peptide sequencing program. *Computer Applications in Biosciences*, 2 :115–118, 1986.

[150] J.M. Hammersley. A few seedlings of research. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statististics and Probabilities*, pages 345–394, Berkeley, California, 1972.

[151] S. Hannenhalli. Polynomial algorithm for computing translocation distance between genomes. In *Sixth Annual Symposium on Combinatorial Pattern Matching*, volume 937 of *Lecture Notes in Computer Science*, pages 162–176, Helsinki, Finland, June 1995. Springer-Verlag.

[152] S. Hannenhalli, C. Chappey, E. Koonin, and P.A. Pevzner. Genome sequence comparison and scenarios for gene rearrangements : A test case. *Genomics*, 30 :299–311, 1995.

[153] S. Hannenhalli and P.A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 581–592, Milwaukee, Wisconsin, 1995.

[154] S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing*, pages 178–189, 1995 (full version appeared in Journal of ACM, 46 : 1–27, 1999).

[155] S. Hannenhalli and P.A. Pevzner. To cut ... or not to cut (applications of comparative physical maps in molecular evolution). In *Seventh Anuual ACM-SIAM Symposium on Discrete Algorithms*, pages 304–313, Atlanta, Georgia, 1996.

[156] S. Hannenhalli, P.A. Pevzner, H. Lewis, S. Skeina, and W. Feldman. Positional sequencing by hybridization. *Computer Applications in Biosciences*, 12 :19–24, 1996.

[157] W.S. Hayes and M. Borodovsky. How to interpret an anonymous bacterial genome : machine learning approach to gene identification. *Genome Research*, 8 :1154–1171, 1998.

[158] S. Henikoff and J.G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences USA*, 89 :10915–10919, 1992.

[159] G.Z. Hertz and G.D. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15 :563–577, 1999.

[160] N. Heuze, S. Olayat, N. Gutman, M.L. Zani, and Y. Courty. Molecular cloning and expression of an alternative hKLK3 transcript coding for a variant protein of prostate-specific antigen. *Cancer Research*, 59 :2820–2824, 1999.

[161] M. H. Heydari and I. H. Sudborough. On the diameter of the pancake network. *Journal of Algorithms*, 25 :67–94, 1997.

[162] D.G. Higgins, J.D. Thompson, and T.J. Gibson. Using CLUSTAL for multiple sequence alignments. *Methods in Enzymology*, 266 :383–402, 1996.

[163] D.S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Communication of ACM*, 18 :341–343, 1975.

[164] D.S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of ACM*, 24 :664–675, 1977.

[165] J.D. Hoheisel, E. Maier, R. Mott, L. McCarthy, A.V. Grigoriev, L.C. Schalkwyk, D. Nizetic, F. Francis, and H. Lehrach. High resolution cosmid and P1 maps spanning the 14 Mb genome of the fission yeast *S. pombe*. *Cell*, 73 :109–120, 1993.

[166] S. Hopper, R.S. Johnson, J.E. Vath, and K. Biemann. Glutaredoxin from rabbit bone marrow. *Journal of Biological Chemistry*, 264 :20438–20447, 1989.

[167] Y. Hu, L.R. Tanzer, J. Cao, C.D. Geringer, and R.E. Moore. Use of long RT-PCR to characterize splice variant mRNAs. *Biotechniques*, 25 :224–229, 1998.

[168] X. Huang, R.C. Hardison, and W. Miller. A space-efficient algorithm for local similarities. *Computer Applications in Biosciences*, 6 :373–381, 1990.

[169] X. Huang and W. Miller. A time-efficient, linear-space local similarity algorithm. *Advances in Applied Mathematics*, 12 :337–357, 1991.

[170] T.J. Hubbard, A.M. Lesk, and A. Tramontano. Gathering them into the fold. *Nature Structural Biology*, 4 :313, 1996.

[171] E. Hubbell. Multiplex sequencing by hybridization. *Journal of Computational Biology*, 8, 2000.

[172] E. Hubbell and P.A. Pevzner. Fidelity probes for DNA arrays. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 113–117, Heidelberg, Germany, August 1999. AAAI Press.

[173] T.J. Hudson, L.D. Stein, S.S. Gerety, J. Ma, A.B. Castle, J. Silva, D.K. Slonim, R. Baptista, L. Kruglyak, S.H. Xu, X. Hu, A.M.E. Colbert, C. Rosenberg, M.P. Reeve-Daly, S. Rozen, L. Hui, X. Wu, C. Vestergaard, K.M. Wilson, and J.S. Bae et al. An STS-based map of the human genome. *Science*, 270 :1945–1954, 1995.

[174] J.W. Hunt and T.G. Szymanski. A fast algorithm for computing longest common subsequences. *Communication of ACM*, 20 :350–353, 1977.

[175] R.M. Idury and M.S. Waterman. A new algorithm for DNA sequence assembly. *Journal of Computational Biology*, 2 :291–306, 1995.

[176] C. Iseli, C.V. Jongeneel, and P. Bucher. ESTScan : a program for detecting, evaluating and reconstructing potential coding regions in EST sequences. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 138–148, Heidelberg, Germany, August 6-10 1999. AAAI Press.

[177] A.G. Ivanov. Distinguishing an approximate word's inclusion on Turing machine in real time. *Izvestiia Academii Nauk SSSR, Series Math.*, 48 :520–568, 1984.

[178] A. Jauch, J. Wienberg, R. Stanyon, N. Arnold, S. Tofanelli, T. Ishida, and T. Cremer. Reconstruction of genomic rearrangements in great apes gibbons by chromosome painting. *Proceedings of the National Academy of Sciences USA*, 89 :8611–8615, 1992.

[179] T. Jiang and R.M. Karp. Mapping clones with a given ordering or interleaving. *Algorithmica*, 21 :262–284, 1998.

[180] T. Jiang and M. Li. Approximating shortest superstrings with constraints. *Theoretical Computer Science*, 134 :473–491, 1994.

[181] T. Jiang and M. Li. DNA sequencing and string learning. *Mathematical Systems Theory*, 29 :387–405, 1996.

[182] R.J. Johnson and K. Biemann. Computer program (SEQPEP) to aid in the interpretation of high-energy collision tandem mass spectra of peptides. *Biomedical and Environmental Mass Spectrometry*, 18 :945–957, 1989.

[183] Y.W. Kan and A.M. Dozy. Polymorphism of DNA sequence adjacent to human beta-globin structural gene : relationship to sickle mutation. *Proceedings of the National Academy of Sciences USA*, 75 :5631–5635, 1978.

[184] S.K. Kannan and E.W. Myers. An algorithm for locating nonoverlapping regions of maximum alignment score. *SIAM Journal on Computing*, 25 :648–662, 1996.

[185] H. Kaplan, R. Shamir, and R.E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 344–351, New Orleans, Louisiana, January 1997.

[186] S. Karlin and S.F. Altschul. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences USA*, 87 :2264–2268, 1990.

[187] S. Karlin and G. Ghandour. Multiple-alphabet amino acid sequence comparisons of the immunoglobulin kappa-chain constant domain. *Proceedings of the National Academy of Sciences USA*, 82 :8597–8601, 1985.

[188] R.M. Karp, R.E. Miller, and A.L. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, pages 125–136, Denver, Colorado, May 1972.

[189] R.M. Karp and M.O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31 :249–260, 1987.

[190] R.M. Karp and R. Shamir. Algorithms for optical mapping. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB-98)*, pages 117–124, New York, New York, March 1998. ACM Press.

[191] J. Kececioglu and R. Ravi. Of mice and men : Evolutionary distances between genomes under translocation. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 604–613, New York, New York, 1995.

[192] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for the reversal distance between two permutations. In *Fourth Annual Symposium on Combinatorial Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, pages 87–105, Padova, Italy, 1993. Springer-Verlag.

[193] J. Kececioglu and D. Sankoff. Efficient bounds for oriented chromosome inversion distance. In *Fifth Annual Symposium on Combinatorial Pattern Matching*, volume 807 of *Lecture Notes in Computer Science*, pages 307–325, Asilomar, California, 1994. Springer-Verlag.

[194] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for the inversion distance between two permutations. *Algorithmica*, 13 :180–210, 1995.

[195] J.D. Kececioglu and E.W. Myers. Combinatorial algorithms for DNA sequence assembly. *Algorithmica*, 13 :7–51, 1995.

[196] T.J. Kelly and H.O. Smith. A restriction enzyme from *Hemophilus influenzae*. II. Base sequence of the recognition site. *Journal of Molecular Biology*, 51 :393–409, 1970.

[197] K. Khrapko, Y. Lysov, A. Khorlin, V. Shik, V. Florent'ev, and A. Mirzabekov. An oligonucleotide hybridization approach to DNA sequencing. *FEBS Letters*, 256 :118–122, 1989.

[198] J.F.C. Kingman. Subadditive ergodic theory. *Annals of Probability*, 6 :883–909, 1973.

[199] J. Kleffe and M. Borodovsky. First and second moment of counts of words in random texts generated by Markov chains. *Computer Applications in Biosciences*, 8 :433–441, 1992.

[200] M. Knill, W.J. Bruno, and D.C. Torney. Non-adaptive group testing in the presence of errors. *Discrete Applied Mathematics*, 88 :261–290, 1998.

[201] D.E. Knuth. Permutations, matrices and generalized Young tableaux. *Pacific Journal of Mathematics*, 34 :709–727, 1970.

[202] D.E. Knuth. *The Art of Computer Programming*, chapter 2. Addison-Wesley, second edition, 1973.

[203] D.E. Knuth, J.H. Morris, and V.R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6 :323–350, 1977.

[204] Y. Kohara, K. Akiyama, and K. Isono. The physical map of the whole E. coli chromosome : application of a new strategy for rapid analysis and sorting of a large genomic library. *Cell*, 50 :495–508, 1987.

[205] G.A. Komatsoulis and M.S. Waterman. Chimeric alignment by dynamic programming : Algorithm and biological uses. In *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97)*, pages 174–180, Santa Fe, New Mexico, January 1997. ACM Press.

[206] A. Kotzig. Moves without forbidden transitions in a graph. *Matematicky Casopis*, 18 :76–80, 1968.

[207] R.G. Krishna and F. Wold. Posttranslational modifications. In R.H. Angeletti, editor, *Proteins - Analysis and Design*, pages 121–206. Academic Press, 1998.

[208] A. Krogh, M. Brown, I.S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology : Applications to protein modeling. *Journal of Molecular Biology*, 235 :1501–1531, 1994.

[209] A. Krogh, I.S. Mian, and D. Haussler. A Hidden Markov Model that finds genes in E. coli DNA. *Nucleic Acids Research*, 22 :4768–4778, 1994.

[210] S. Kruglyak. Multistage sequencing by hybridization. *Journal of Computational Biology*, 5 :165–171, 1998.

[211] J.B. Kruskal and D. Sankoff. An anthology of algorithms and concepts for sequence comparison. In D. Sankoff and J.B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules : The Theory and Practice of Sequence Comparison*, pages 265–310. Addison-Wesley, 1983.

[212] G.M. Landau and J.P. Schmidt. An algorithm for approximate tandem repeats. In *Fourth Annual Symposium on Combinatorial Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, pages 120–133, Padova, Italy, 2-4 June 1993. Springer-Verlag.

[213] G.M. Landau and U. Vishkin. Efficient string matching in the presence of errors. In *26th Annual Symposium on Foundations of Computer Science*, pages 126–136, Los Angeles, California, October 1985.

[214] E.S. Lander and M.S. Waterman. Genomic mapping by fingerprinting random clones : a mathematical analysis. *Genomics*, 2 :231–239, 1988.

[215] K. Lange, M. Boehnke, D.R. Cox, and K.L. Lunetta. Statistical methods for polyploid radiation hybrid mapping. *Genome Research*, 5 :136–150, 1995.

[216] E. Lawler and S. Sarkissian. Adaptive error correcting codes based on cooperative play of the game of “Twenty Questions Game with a Liar". In *Proceedings of Data Compression Conference DCC ‘95*, page 464, Los Alamitos, California, 1995. IEEE Computer Society Press.

[217] C.E. Lawrence, S.F. Altschul, M.S. Boguski, J.S. Liu, A.F. Neuwald, and J.C. Wootton. Detecting subtle sequence signals : a Gibbs sampling strategy for multiple alignment. *Science*, 262 :208–214, October 1993.

[218] J.K. Lee, V. Dancik, and M.S. Waterman. Estimation for restriction sites observed by optical mapping using reversible-jump Markov chain Monte Carlo. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB-98)*, pages 147–152, New York, New York, March 1998. ACM Press.

[219] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 6 :707–710, 1966.

[220] B. Lewin. *Genes VII.* Oxford University Press, 1999.

[221] M. Li, B. Ma, and L. Wang. Finding similar regions in many strings. In *Proceedings of the 31st ACM Annual Symposium on Theory of Computing*, pages 473–482, Atlanta, Georgia, May 1999.

[222] S.Y.R. Li. A martingale approach to the study of ocurrence of sequence patterns in repeated experiments. *Annals of Probability*, 8 :1171–1176, 1980.

[223] J. Lingner, T.R. Hughes, A. Shevchenko, M. Mann, V. Lundblad, and T.R. Cech. Reverse transcriptase motifs in the catalytic subunit of telomerase. *Science*, 276 :561–567, 1997.

[224] D.J. Lipman, S. F Altschul, and J.D. Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences USA*, 86 :4412–4415, 1989.

[225] D.J. Lipman and W.R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227 :1435–1441, 1985.

[226] R.J. Lipshutz, D. Morris, M. Chee, E. Hubbell, M.J. Kozal, N. Shah, N. Shen, R. Yang, and S.P.A. Fodor. Using oligonucleotide probe arrays to access genetic diversity. *Biotechniques*, 19 :442–447, 1995.

[227] B.F. Logan and L.A. Shepp. A variational problem for random Young tableaux. *Advances in Mathematics*, 26 :206–222, 1977.

[228] Y. Lysov, V. Florent'ev, A. Khorlin, K. Khrapko, V. Shik, and A. Mirzabekov. DNA sequencing by hybridization with oligonucleotides. *Doklady Academy Nauk USSR*, 303 :1508–1511, 1988.

[229] C.A. Makaroff and J.D. Palmer. Mitochondrial DNA rearrangements and transcriptional alterations in the male sterile cytoplasm of Ogura radish. *Molecular Cellular Biology*, 8 :1474–1480, 1988.

[230] M. Mann and M. Wilm. Error-tolerant identification of peptides in sequence databases by peptide sequence tags. *Analytical Chemistry*, 66 :4390–4399, 1994.

[231] M. Mann and M. Wilm. Electrospray mass-spectrometry for protein characterization. *Trends in Biochemical Sciences*, 20 :219–224, 1995.

[232] D. Margaritis and S.S. Skiena. Reconstructing strings from substrings in rounds. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 613–620, Los Alamitos, California, October 1995.

[233] A.M. Maxam and W. Gilbert. A new method for sequencing DNA. *Proceedings of the National Academy of Sciences USA*, 74 :560–564, 1977.

[234] G. Mayraz and R. Shamir. Construction of physical maps from oligonucleotide fingerprints data. *Journal of Computational Biology*, 6 :237–252, 1999.

[235] F.R. McMorris, C. Wang, and P. Zhang. On probe interval graphs. *Discrete Applied Mathematics*, 88 :315–324, 1998.

[236] W. Miller and E.W. Myers. Sequence comparison with concave weighting functions. *Bulletin of Mathematical Biology*, 50 :97–120, 1988.

[237] A. Milosavljevic and J. Jurka. Discovering simple DNA sequences by the algorithmic significance method. *Computer Applications in Biosciences*, 9 :407–411, 1993.

[238] B. Mirkin and F.S. Roberts. Consensus functions and patterns in molecular sequences. *Bulletin of Mathematical Biology*, 55 :695–713, 1993.

[239] A.A. Mironov and N.N. Alexandrov. Statistical method for rapid homology search. *Nucleic Acids Research*, 16 :5169–5174, 1988.

[240] A.A. Mironov, J.W. Fickett, and M.S. Gelfand. Frequent alternative splicing of human genes. *Genome Research*, 9 :1288–1293, 1999.

[241] A.A. Mironov and P.A. Pevzner. SST versus EST in gene recognition. *Microbial and Comparative Genomics*, 4 :167–172, 1999.

[242] A.A. Mironov, M.A. Roytberg, P.A. Pevzner, and M.S. Gelfand. Performance guarantee gene predictions via spliced alignment. *Genomics*, 51 :332–339, 1998.

[243] S. Muthukrishnan and L. Parida. Towards constructing physical maps by optical mapping : An effective, simple, combinatorial approach. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97)*, pages 209–219, Santa Fe, New Mexico, January 1997. ACM Press.

[244] M. Muzio, A.M. Chinnaiyan, F.C. Kischkel, K. O'Rourke, A. Shevchenko, J. Ni, C. Scaffidi, J.D. Bretz, M. Zhang, R. Gentz, M. Mann, P.H. Krammer, M.E. Peter, and V.M. Dixit. FLICE, a novel FADD-homologous ICE/CED-3-like protease, is recruited to the CD95 (Fas/APO-1) death-inducing signaling complex. *Cell*, 85 :817–827, 1996.

[245] E.W. Myers. A sublinear algorithm for approximate keyword searching. *Algorithmica*, 12 :345–374, 1994.

[246] E.W. Myers and W. Miller. Optimal alignments in linear space. *Computer Applications in Biosciences*, 4 :11–17, 1988.

[247] G. Myers. Whole genome shotgun sequencing. *IEEE Computing in Science and Engineering*, 1 :33–43, 1999.

[248] J.H. Nadeau and B.A. Taylor. Lengths of chromosomal segments conserved since divergence of man and mouse. *Proceedings of the National Academy of Sciences USA*, 81 :814–818, 1984.

[249] K. Nakata, M. Kanehisa, and C. DeLisi. Prediction of splice junctions in mRNA sequences. *Nucleic Acids Research*, 13 :5327–5340, 1985.

[250] D. Naor and D. Brutlag. On near-optimal alignments of biological sequences. *Journal of Computational Biology*, 1 :349–366, 1994.

[251] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48 :443–453, 1970.

[252] L. Newberg and D. Naor. A lower bound on the number of solutions to the probed partial digest problem. *Advances in Applied Mathematics*, 14 :172–183, 1993.

[253] R. Nussinov, G. Pieczenik, J.R. Griggs, and D.J. Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied Mathematics*, 35 :68–82, 1978.

[254] S. O'Brien and J. Graves. Report of the committee on comparative gene mapping in mammals. *Cytogenetics and Cell Genetics*, 58 :1124–1151, 1991.

[255] S. Ohno. *Sex chromosomes and sex-linked genes*. Springer-Verlag, 1967.

[256] S. Ohno, U. Wolf, and N.B. Atkin. Evolution from fish to mammals by gene duplication. *Hereditas*, 59 :708–713, 1968.

[257] M.V. Olson, J.E. Dutchik, M.Y. Graham, G.M. Brodeur, C. Helms, M. Frank, M. MacCollin, R. Scheinman, and T. Frank. Random-clone strategy for genomic restriction mapping in yeast. *Proceedings of the National Academy of Sciences USA*, 83 :7826–7830, 1986.

[258] O. Owolabi and D.R. McGregor. Fast approximate string matching. *Software Practice and Experience*, 18 :387–393, 1988.

[259] J.D. Palmer and L.A. Herbon. Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 27 :87–97, 1988.

[260] A.H. Paterson, T.H. Lan, K.P. Reischmann, C. Chang, Y.R. Lin, S.C. Liu, M.D. Burow, S.P. Kowalski, C.S. Katsar, T.A. DelMonte, K.A. Feldmann, K.F. Schertz, and J.F. Wendel. Toward a unified genetic map of higher plants, transcending the monocot-dicot divergence. *Nature Genetics*, 15 :380–382, 1996.

[261] S.D. Patterson and R. Aebersold. Mass spectrometric approaches for the identification of gel-separated proteins. *Electrophoresis*, 16 :1791–1814, 1995.

[262] H. Peltola, H. Soderlund, and E. Ukkonen. SEQAID : a DNA sequence assembling program based on a mathematical model. *Nucleic Acids Research*, 12 :307–321, 1984.

[263] M. Perlin and A. Chakravarti. Efficient construction of high-resolution physical maps from yeast artificial chromosomes using radiation hybrids : inner product mapping. *Genomics*, 18 :283–289, 1993.

[264] P.A. Pevzner. l-tuple DNA sequencing : computer analysis. *Journal of Biomolecular Structure and Dynamics*, 7 :63–73, 1989.

[265] P.A. Pevzner. Multiple alignment, communication cost, and graph matching. *SIAM Journal on Applied Mathematics*, 52 :1763–1779, 1992.

[266] P.A. Pevzner. Statistical distance between texts and filtration methods in rapid similarity search algorithm. *Computer Applications in Biosciences*, 8 :121–27, 1992.

[267] P.A. Pevzner. DNA physical mapping and alternating Eulerian cycles in colored graphs. *Algorithmica*, 13 :77–105, 1995.

[268] P.A. Pevzner. DNA statistics, overlapping word paradox and Conway equation. In H.A. Lim, J.W. Fickett, C.R. Cantor, and R.J. Robbins, editors, *Proceedings of the Second International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis*, pages 61–68, St. Petersburg Beach, Florida, June 1993. World Scientific.

[269] P.A. Pevzner, M.Y. Borodovsky, and A.A. Mironov. Linguistics of nucleotide sequences. I : The significance of deviations from mean statistical characteristics and prediction of the frequencies of occurrence of words. *Journal of Biomolecular Structure and Dynamics*, 6 :1013–1026, 1989.

[270] P.A. Pevzner, V. Dancik, and C.L. Tang. Mutation-tolerant protein identification by mass-spectrometry. In R. Shamir, S. Miyano, S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology (RECOMB-00)*, pages 231–236, Tokyo, Japan, April 2000. ACM Press.

[271] P.A. Pevzner and R. Lipshutz. Towards DNA sequencing chips. In *Proceedings of the 19th International Conference on Mathematical Foundations of Computer Science*, volume 841 of *Lecture Notes in Computer Science*, pages 143–158, Kosice, Slovakia, 1994.

[272] P.A. Pevzner, Y. Lysov, K. Khrapko, A. Belyavski, V. Florentiev, and A. Mirzabekov. Improved chips for sequencing by hybridization. *Journal of Biomolecular Structure and Dynamics*, 9 :399–410, 1991.

[273] P.A. Pevzner and M.S. Waterman. Generalized sequence alignment and duality. *Advances in Applied Mathematics*, 14(2) :139–171, 1993.

[274] P.A. Pevzner and M.S. Waterman. Multiple filtration and approximate pattern matching. *Algorithmica*, 13 :135–154, 1995.

[275] P.A. Pevzner and M.S. Waterman. Open combinatorial problems in computational molecular biology. In *Third Israeli Symposium on the Theory of Computing and Systems*, Tel-Aviv, Israel, January 1995.

[276] S. Pilpel. Descending subsequences of random permutations. *Journal of Combinatorial Theory, Series A*, 53 :96–116, 1990.

[277] A. Pnueli, A. Lempel, and S. Even. Transitive orientation of graphs and identification of permutation graphs. *Canadian Journal of Mathematics*, 23 :160–175, 1971.

[278] J.H. Postlethwait, Y.L. Yan, M.A. Gates, S. Horne, A. Amores, A. Brownlie, A. Donovan, E.S. Egan, A. Force, Z. Gong, C. Goutel, A. Fritz, R. Kelsh, E. Knapik, E. Liao, B. Paw, D. Ransom, A. Singer, M. Thomson, T.S. Abduljabbar, P. Yelick, D. Beier, J.S. Joly, D. Larhammar, and F. Rosa et al. Vertebrate genome evolution and the zebrafish gene map. *Nature Genetics*, 345-349 :18, 1998.

[279] A. Poustka, T. Pohl, D.P. Barlow, G. Zehetner, A. Craig, F. Michiels, E. Ehrich, A.M. Frischauf, and H. Lehrach. Molecular approaches to mammalian genetics. *Cold Spring Harbor Symposium on Quantitative Biology*, 51 :131–139, 1986.

[280] F.P. Preparata, A.M. Frieze, and E. Upfal. On the power of universal bases in sequencing by hybridization. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Third Annual International Conference on Computational Molecular Biology (RECOMB-99)*, pages 295–301, Lyon, France, April 1999. ACM Press.

[281] B. Prum, F. Rudolphe, and E. De Turckheim. Finding words with unexpected frequences in DNA sequences. *Journal of Royal Statistical Society, Series B*, 57 :205–220, 1995.

[282] M. Regnier and W. Szpankowski. On the approximate pattern occurrences in a text. In *Compression and Complexity of Sequences 1997*, pages 253–264, 1998.

[283] K. Reinert, H.-P. Lenhof, P. Mutzel, K. Mehlhorn, and J.D. Kececioglu. A branch-and-cut algorithm for multiple sequence alignment. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97)*, pages 241–250, Santa Fe, New Mexico, January 1997. ACM Press.

[284] G. Rettenberger, C. Klett, U. Zechner, J. Kunz, W. Vogel, and H. Hameister. Visualization of the conservation of synteny between humans and pigs by hetereologous chromosomal painting. *Genomics*, 26 :372–378, 1995.

[285] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences. *Bioinformatics*, 14 :55–67, 1998.

[286] J.C. Roach, C. Boysen, K. Wang, and L. Hood. Pairwise end sequencing : a unified approach to genomic mapping and sequencing. *Genomics*, 26 :345–353, 1995.

[287] G.de E. Robinson. On representations of the symmetric group. *American Journal of Mathematics*, 60 :745–760, 1938.

[288] E. Rocke and M. Tompa. An algorithm for finding novel gapped motifs in DNA sequences. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB-98)*, pages 228–233, New York, New York, March 1998. ACM Press.

[289] J. Rosenblatt and P.D. Seymour. The structure of homometric sets. *SIAM Journal on Alg. Discrete Methods*, 3 :343–350, 1982.

[290] M.A. Roytberg. A search for common pattern in many sequences. *Computer Applications in Biosciences*, 8 :57–64, 1992.

[291] A.R. Rubinov and M.S. Gelfand. Reconstruction of a string from substring precedence data. *Journal of Computational Biology*, 2 :371–382, 1995.

[292] B.E. Sagan. *The Symmetric Group : Representations, Combinatorial Algorithms, and Symmetric Functions*. Wadsworth Brooks Cole Mathematics Series, 1991.

[293] M.F. Sagot, A. Viari, and H. Soldano. Multiple sequence comparison—a peptide matching approach. *Theoretical Computer Science*, 180 :115–137, 1997.

[294] T. Sakurai, T. Matsuo, H. Matsuda, and I. Katakuse. PAAS 3 : A computer program to determine probable sequence of peptides from mass spectrometric data. *Biomedical Mass Spectrometry*, 11 :396–399, 1984.

[295] S.L. Salzberg, A.L. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26 :544–548, 1998.

[296] S.L. Salzberg, D.B. Searls, and S. Kasif. *Computational Methods in Molecular Biology*. Elsevier, 1998.

[297] F. Sanger, S. Nilken, and A.R. Coulson. DNA sequencing with chain terminating inhibitors. *Proceedings of the National Academy of Sciences USA*, 74 :5463–5468, 1977.

[298] D. Sankoff. Minimum mutation tree of sequences. *SIAM Journal on Applied Mathematics*, 28 :35–42, 1975.

[299] D. Sankoff. Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM Journal on Applied Mathematics*, 45 :810–825, 1985.

[300] D. Sankoff. Edit distance for genome comparison based on non-local operations. In *Third Annual Symposium on Combinatorial Pattern Matching*, volume 644 of *Lecture Notes in Computer Science*, pages 121–135, Tucson, Arizona, 1992. Springer-Verlag.

[301] D. Sankoff and M. Blanchette. Multiple genome rearrangements. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB-98)*, pages 243–247, New York, New York, March 1998. ACM Press.

[302] D. Sankoff, R. Cedergren, and Y. Abel. Genomic divergence through gene rearrangement. In *Molecular Evolution : Computer Analysis of Protein and Nucleic Acid Sequences*, chapter 26, pages 428–438. Academic Press, 1990.

[303] D. Sankoff and M. Goldstein. Probabilistic models of genome shuffling. *Bulletin of Mathematical Biology*, 51 :117–124, 1989.

[304] D. Sankoff, G. Leduc, N. Antoine, B. Paquin, B. Lang, and R. Cedergren. Gene order comparisons for phylogenetic inference : Evolution

of the mitochondrial genome. *Proceedings of the National Academy of Sciences USA*, 89 :6575–6579, 1992.

[305] D. Sankoff and S. Mainville. Common subsequences and monotone subsequences. In D. Sankoff and J.B. Kruskal, editors, *Time Warps, String Edits, and Macromolecules : The Theory and Practice of Sequence Comparison*, pages 363–365. Addison-Wesley, 1983.

[306] C. Schensted. Longest increasing and decreasing subsequences. *Canadian Journal of Mathematics*, 13 :179–191, 1961.

[307] H. Scherthan, T. Cremer, U. Arnason, H. Weier, A. Lima de Faria, and L. Fronicke. Comparative chromosomal painting discloses homologous segments in distantly related mammals. *Nature Genetics*, 6 :342–347, 1994.

[308] J.P. Schmidt. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM Journal on Computing*, 27 :972–992, 1998.

[309] W. Schmitt and M.S. Waterman. Multiple solutions of DNA restriction mapping problem. *Advances in Applid Mathematics*, 12 :412–427, 1991.

[310] M. Schoniger and M.S. Waterman. A local algorithm for DNA sequence alignment with inversions. *Bulletin of Mathematical Biology*, 54 :521–536, 1992.

[311] D.C. Schwartz, X. Li, L.I. Hernandez, S.P. Ramnarain, E.J. Huff, and Y.K. Wang. Ordered restriction maps of Saccharomyces cerevisiae chromosomes constructed by optical mapping. *Science*, 262 :110–114, 1993.

[312] D. Searls and S. Dong. A syntactic pattern recognition system for DNA sequences. In H.A. Lim, J.W. Fickett, C.R. Cantor, and R.J. Robbins, editors, *Proceedings of the Second International Conference on Bioinformatics, Supercomputing, and Complex Genome Analysis*, pages 89–102, St. Petersburg Beach, Florida, June 1993. World Scientific.

[313] D. Searls and K. Murphy. Automata-theoretic models of mutation and alignment. In *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 341–349, Cambridge, England, 1995.

[314] S.S. Skiena, W.D. Smith, and P. Lemke. Reconstructing sets from interpoint distances. In *Proceedings of Sixth Annual Symposium on Computational Geometry*, pages 332–339, Berkeley, California, June, 1990.

[315] S.S. Skiena and G. Sundaram. A partial digest approach to restriction site mapping. *Bulletin of Mathematical Biology*, 56 :275–294, 1994.

[316] S.S. Skiena and G. Sundram. Reconstructing strings from substrings. *Journal of Computational Biology*, 2 :333–354, 1995.

[317] D. Slonim, L. Kruglyak, L. Stein, and E. Lander. Building human genome maps with radiation hybrids. In S. Istrail, P.A. Pevzner, and M.S.

Waterman, editors, *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97)*, pages 277–286, Santa Fe, New Mexico, January 1997. ACM Press.

[318] H.O. Smith, T.M. Annau, and S. Chandrasegaran. Finding sequence motifs in groups of functionally related proteins. *Proceedings of the National Academy of Sciences USA*, 87 :826–830, 1990.

[319] H.O. Smith and K.W. Wilcox. A restriction enzyme from Hemophilus influenzae. I. Purification and general properties. *Journal of Molecular Biology*, 51 :379–391, 1970.

[320] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147 :195–197, 1981.

[321] E.E. Snyder and G.D. Stormo. Identification of coding regions in genomic DNA sequences : an application of dynamic programming and neural networks. *Nucleic Acids Research*, 21 :607–613, 1993.

[322] E.E. Snyder and G.D. Stormo. Identification of protein coding regions in genomic DNA. *Journal of Molecular Biology*, 248 :1–18, 1995.

[323] V.V. Solovyev, A.A. Salamov, and C.B. Lawrence. Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. *Nucleic Acids Research*, 22 :5156–63, 1994.

[324] E.L. Sonnhammer, S.R. Eddy, and R. Durbin. Pfam : a comprehensive database of protein domain families based on seed alignments. *Proteins*, 28 :405–420, 1997.

[325] E. Southern. United Kingdom patent application GB8810400. 1988.

[326] R. Staden. Methods for discovering novel motifs in nucleic acid seqences. *Computer Applications in Biosciences*, 5 :293–298, 1989.

[327] R. Staden and A.D. McLachlan. Codon preference and its use in identifying protein coding regions in long DNA sequences. *Nucleic Acids Research*, 10 :141–156, 1982.

[328] J.M. Steele. An Efron-Stein inequality for nonsymmetric statistics. *Annals of Statistics*, 14 :753–758, 1986.

[329] M. Stefik. Inferring DNA structure from segmentation data. *Artificial Intelligence*, 11 :85–144, 1978.

[330] E.E. Stuckle, C. Emmrich, U. Grob, and P.J. Nielsen. Statistical analysis of nucleotide sequences. *Nucleic Acids Research*, 18 :6641–6647, 1990.

[331] A.H. Sturtevant and T. Dobzhansky. Inversions in the third chromosome of wild races of *Drosophila pseudoobscura*, and their use in the study of the history of the species. *Proceedings of the National Academy of Sciences USA*, 22 :448–450, 1936.

[332] S.H. Sze and P.A. Pevzner. Las Vegas algorithms for gene recognition : subotimal and error tolerant spliced alignment. *Journal of Computational Biology*, 4 :297–310, 1997.

[333] J. Tarhio and E. Ukkonen. A greedy approximation algorithm for constructing shortest common superstrings. *Theoretical Computer Science*, 57 :131–145, 1988.

[334] J. Tarhio and E. Ukkonen. Boyer-Moore approach to approximate string matching. In J.R. Gilbert and R. Karlsson, editors, *Proceedings of the Second Scandinavian Workshop on Algorithm Theory*, number 447 in Lecture Notes in Computer Science, pages 348–359, Bergen, Norway, 1990. Springer-Verlag.

[335] J.A. Taylor and R.S. Johnson. Sequence database searches via *de novo* peptide sequencing by tandem mass spectrometry. *Rapid Communications in Mass Spectrometry*, 11 :1067–1075, 1997.

[336] W.R. Taylor. Multiple sequence alignment by a pairwise algorithm. *Computer Applications in Biosciences*, 3 :81–87, 1987.

[337] S.M. Tilghman, D.C. Tiemeier, J.G. Seidman, B.M. Peterlin, M. Sullivan, J.V. Maizel, and P. Leder. Intervening sequence of DNA identified in the structural portion of a mouse beta-globin gene. *Proceedings of the National Academy of Sciences USA*, 75 :725–729, 1978.

[338] M. Tompa. An exact method for finding short motifs in sequences with application to the Ribosome Binding Site problem. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 262–271, Heidelberg, Germany, August 1999. AAAI Press.

[339] E. Uberbacher and R. Mural. Locating protein coding regions in human DNA sequences by a multiple sensor - neural network approach. *Proceedings of the National Academy of Sciences USA*, 88 :11261–11265, 1991.

[340] E. Ukkonen. Approximate string matching with q-grams and maximal matches. *Theoretical Computer Science*, 92 :191–211, 1992.

[341] S. Ulam. Monte-Carlo calculations in problems of mathematical physics. In *Modern mathematics for the engineer*, pages 261–281. McGraw-Hill, 1961.

[342] A.M. Vershik and S.V. Kerov. Asymptotics of the Plancherel measure of the symmetric group and the limiting form of Young tableaux. *Soviet Mathematical Doklady*, 18 :527–531, 1977.

[343] M. Vihinen. An algorithm for simultaneous comparison of several sequences. *Computer Applications in Biosciences*, 4 :89–92, 1988.

[344] M. Vingron and P. Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *Journal of Molecular Biology*, 218 :33–43, 1991.

[345] M. Vingron and P.A. Pevzner. Multiple sequence comparison and consistency on multipartite graphs. *Advances in Applied Mathematics*, 16 :1–22, 1995.

[346] M. Vingron and M.S. Waterman. Sequence alignment and penalty choice. Review of concepts, studies and implications. *Journal of Molecular Biology*, 235 :1–12, 1994.

[347] T.K. Vintsyuk. Speech discrimination by dynamic programming. *Comput.*, 4 :52–57, 1968.

[348] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13 :260–269, 1967.

[349] D.G. Wang, J.B. Fan, C.J. Siao, A. Berno, P. Young, R. Sapolsky, G. Ghandour, N. Perkins, E. Winchester, J. Spencer, L. Kruglyak, L. Stein, L. Hsie, T. Topaloglou, E. Hubbell, E. Robinson, M. Mittmann, M.S. Morris, N. Shen, D. Kilburn, J. Rioux, C. Nusbaum, S. Rozen, T.J. Hudson, and E.S. Lander et al. Large-scale identification, mapping, and genotyping of single-nucleotide polymorphisms in the human genome. *Science*, 280 :1074–1082, 1998.

[350] L. Wang and D. Gusfield. Improved approximation algorithms for tree alignment. In *Seventh Annual Symposium on Combinatorial Pattern Matching*, volume 1075 of *Lecture Notes in Computer Science*, pages 220–233, Laguna Beach, California, 10-12 June 1996. Springer-Verlag.

[351] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1 :337–348, 1994.

[352] L. Wang, T. Jiang, and E.L. Lawler. Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica*, 16 :302–315, 1996.

[353] M.D. Waterfield, G.T. Scrace, N. Whittle, P. Stroobant, A. Johnsson, A. Wasteson, B. Westermark, C.H. Heldin, J.S. Huang, and T.F. Deuel. Platelet-derived growth factor is structurally related to the putative transforming protein p28sis of simian sarcoma virus. *Nature*, 304 :35–39, 1983.

[354] M.S. Waterman. Secondary structure of single-stranded nucleic acids. *Studies in Foundations and Combinatorics, Advances in Mathematics Supplementary Studies*, 1 :167–212, 1978.

[355] M.S. Waterman. Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. *Proceedings of the National Academy of Sciences USA*, 80 :3123–3124, 1983.

[356] M.S. Waterman. Efficient sequence alignment algorithms. *Journal of Theoretical Biology*, 108 :333–337, 1984.

[357] M.S. Waterman. *Introduction to Computational Biology.* Chapman Hall, 1995.

[358] M.S. Waterman, R. Arratia, and D.J. Galas. Pattern recognition in several sequences : consensus and alignment. *Bulletin of Mathematical Biology*, 46 :515–527, 1984.

[359] M.S. Waterman and M. Eggert. A new algorithm for best subsequence alignments with application to tRNA–rRNA comparisons. *Journal of Molecular Biology*, 197 :723–728, 1987.

[360] M.S. Waterman, M. Eggert, and E. Lander. Parametric sequence comparisons. *Proceedings of the National Academy of Sciences USA*, 89 :6090–6093, 1992.

[361] M.S. Waterman and J.R. Griggs. Interval graphs and maps of DNA. *Bulletin of Mathematical Biology*, 48 :189–195, 1986.

[362] M.S. Waterman and M.D. Perlwitz. Line geometries for sequence comparisons. *Bulletin of Mathematical Biology*, 46 :567–577, 1984.

[363] M.S. Waterman and T.F. Smith. Rapid dynamic programming algorithms for RNA secondary structure. *Advances in Applied Mathematics*, 7 :455–464, 1986.

[364] M.S. Waterman, T.F. Smith, and W.A. Beyer. Some biological sequence metrics. *Advances in Mathematics*, 20 :367–387, 1976.

[365] M.S. Waterman and M. Vingron. Rapid and accurate estimates of statistical significance for sequence data base searches. *Proceedings of the National Academy of Sciences USA*, 91 :4625–4628, 1994.

[366] G.A. Watterson, W.J. Ewens, T.E. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theoretical Biology*, 99 :1–7, 1982.

[367] J. Weber and G. Myers. Whole genome shotgun sequencing. *Genome Research*, 7 :401–409, 1997.

[368] W.J. Wilbur and D.J. Lipman. Rapid similarity searches of nucleic acid protein data banks. *Proceedings of the National Academy of Sciences USA*, 80 :726–730, 1983.

[369] K.H. Wolfe and D.C. Shields. Molecular evidence for an ancient duplication of the entire yeast genome. *Nature*, 387 :708–713, 1997.

[370] F. Wolfertstetter, K. Frech, G. Herrmann, and T. Werner. Identification of functional elements in unaligned nucleic acid sequences. *Computer Applications in Biosciences*, 12 :71–80, 1996.

[371] S. Wu and U. Manber. Fast text searching allowing errors. *Communication of ACM*, 35 :83–91, 1992.

[372] G. Xu, S.H. Sze, C.P. Liu, P.A. Pevzner, and N. Arnheim. Gene hunting without sequencing genomic clones : finding exon boundaries in cDNAs. *Genomics*, 47 :171–179, 1998.

[373] J. Yates, J. Eng, and A. McCormack. Mining genomes : Correlating tandem mass-spectra of modified and unmodified peptides to sequences in nucleotide databases. *Analytical Chemistry*, 67 :3202–3210, 1995.

[374] J. Yates, J. Eng, A. McCormack, and D. Schieltz. Method to correlate tandem mass spectra of modified peptides to amino acid sequences in the protein database. *Analytical Chemistry*, 67 :1426–1436, 1995.

[375] J. Yates, P. Griffin, L. Hood, and J. Zhou. Computer aided interpretation of low energy MS/MS mass spectra of peptides. In J.J. Villafranca, editor, *Techniques in Protein Chemistry II*, pages 477–485. Academic Press, 1991.

[376] P. Zhang, E.A. Schon, S.G. Fischer, E. Cayanis, J. Weiss, S. Kistler, and P.E. Bourne. An algorithm based on graph theory for the assembly of contigs in physical mapping. *Computer Applications in Biosciences*, 10 :309–317, 1994.

[377] Z. Zhang. An exponential example for a partial digest mapping algorithm. *Journal of Computational Biology*, 1 :235–239, 1994.

[378] D. Zidarov, P. Thibault, M.J. Evans, and M.J. Bertrand. Determination of the primary structure of peptides using fast atom bombardment mass spectrometry. *Biomedical and Environmental Mass Spectrometry*, 19 :13–16, 1990.

[379] R. Zimmer and T. Lengauer. Fast and numerically stable parametric alignment of biosequences. In S. Istrail, P.A. Pevzner, and M.S. Waterman, editors, *Proceedings of the First Annual International Conference on Computational Molecular Biology (RECOMB-97)*, pages 344–353, Santa Fe, New Mexico, January 1997. ACM Press.

[380] M. Zuker. RNA folding. *Methods in Enzymology*, 180 :262–288, 1989.

[381] M. Zuker and D. Sankoff. RNA secondary structures and their prediction. *Bulletin of Mathematical Biology*, 46 :591–621, 1984.

Index



Achevé d'imprimer sur les presses de l'Imprimerie BARNÉOUD
B.P. 44 - 53960 BONCHAMP-LÈS-LAVAL
Dépôt légal : octobre 2006 - N° d'imprimeur : 609064
Imprimé en France